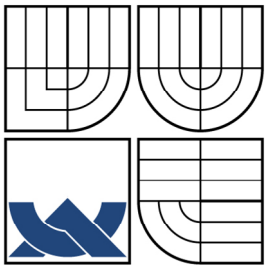
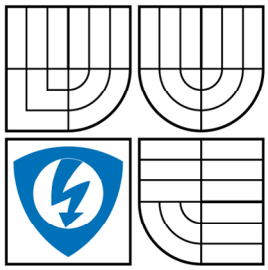


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MIKROPOČÍTAČOVÝ TERMOSTAT I

MICROCONTROLLER ORIENTED THERMOSTAT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JOSEF VALÍČEK
AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. František Zezulka, CSc.
SUPERVISOR

BRNO 2009

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Josef Valíček	***ID:***	72778
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Mikropočítačový termostat I

POKYNY PRO VYPRACOVÁNÍ:

Proveďte ideový návrh mikropočítačově orientovaného pokojového termostatu. V ideovém návrhu zohledněte potřebné funkce a algoritmy řízení a regulace teploty. Navrhněte různé algoritmy řízení a regulace teploty a na matematickém modelu proveďte jejich srovnání. Implementujte vybraný algoritmus do vývojové desky a ověřte výsledek na reálném objektu. Proveďte vyhodnocení měřením.

DOPORUČENÁ LITERATURA:

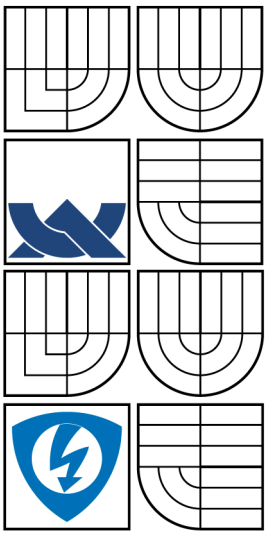
Kubík, Kotek, Strejc, Štecha: Teorie automatického řízení I, SNTL
Valíček J., Paděra Z.: Pokojový termostat nové generace. Semestrální práce. UAMT 2008

Termín zadání:	9.2.2009	***Termín odevzdání:***	1.6.2009

Vedoucí práce: prof. Ing. František Zezulka, CSc.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

MIKROPOČÍTAČOVÝ TERMOSTAT I

Bakalářská práce

Autor: Josef Valíček

Vedoucí: prof. Ing. František Zezulka, CSc.

Abstrakt

Cílem bakalářské práce bylo vypracovat ideový návrh mikropočítačově orientovaného pokojového termostatu, použitelného k regulaci teploty v domech nebo bytech. V návrhu pokojového termostatu jsou zohledněny požadavky na vyráběné termostaty a regulaci teploty v domech. Výsledkem ideového návrhu je elektrické schéma, podle kterého by bylo možné termostat sestavit.

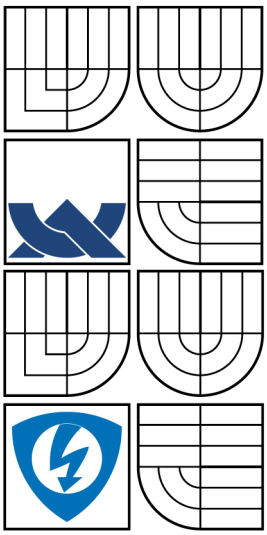
Těžištěm práce je návrh pokročilých algoritmů použitelných pro regulaci teploty v místnostech a srovnání algoritmů na termodynamickém modelu místnosti. Vybraný algoritmus byl implementován do vývojového kitu s mikroprocesorem Rabbit 2000™ a následně byla jeho funkčnost ověřena na reálném objektu.

Posledním bodem práce je vyhodnocení výsledků a výběr nejvhodnějšího algoritmu regulace teploty pro navržený termostat.

Klíčová slova

termostat, ruční ovládání, pomocná regulovaná veličina, ON/OFF regulace

Brno 2009



BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

MICROCONTROLLER ORIENTED THETMOSTAT I

Bachelor's thesis

Author: Josef Valíček

Supervisor: prof. Ing. František Zezulka, CSc.

Abstract

The aim of the bachelor's thesis is to design and develop a microcontroller based room thermostat, suitable for controlling the temperature in houses or flats. The proposal of the thermostat takes into consideration the requirements put on today's marketed thermostats and the regulation of house temperature. The result is a detailed circuit blueprint, suitable for assembling the thermostat device.

The work is mainly focused on the design of advanced algorithms applicable for the room temperature regulation and comparison of these algorithms based on the room thermodynamic model. The selected algorithm was implemented in the Rabbit2000^{TM} microprocessor development kit and its performance was subsequently verified at the real object.

Finally, the thesis gives an evaluation of the results and suggests the most suitable algorithm for the designed thermostat.

Keywords

Thermostat, manual control, auxiliary controlled variable, ON/OFF regulation

Brno 2009

Bibliografická citace

VALÍČEK, J. *Mikropočítačový termostat I.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 62 s. Vedoucí bakalářské práce prof. Ing. František Zezulka, CSc.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Mikropočítačový termostat I jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **1. června 2009** ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Prof. Ing. Františkovi Zezulkovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce

V Brně dne: **1. června 2009** ……
podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Cílem bakalářské práce bylo vypracovat ideový návrh mikropočítačově orientovaného pokojového termostatu, použitelného k regulaci teploty v domech nebo bytech. V úvodní části návrhu pokojového termostatu jsou vypsány požadavky na navrhovaný pokojový termostat. Na základě těchto požadavků jsou vybrány nejvhodnější součástky pro sestavení pokojového termostatu. Výsledkem ideového návrhu je elektrické schéma, podle kterého by bylo možné pokojový termostat vyrobit.

Těžištěm práce je návrh pokročilých algoritmů použitelných pro regulaci teploty v místnostech a srovnání algoritmů na termodynamickém modelu místnosti navrženém v semestrální práci [17]. Z výsledků srovnání algoritmů jsme vybrali nejvhodnější pro regulaci teploty v místnosti.

Dalším bodem práce bylo implementovat vybraný algoritmus do vývojové desky s mikroprocesorem Rabbit 2000^{TM} a následné ověření funkčnosti na reálném objektu.

Závěrem práce je provedeno vyhodnocení výsledků podle všech hledisek regulace teploty v místnostech.

2. NÁVRH TERMOSTATU

2.1 POŽADAVKY NA POKOJOVÝ TERMOSTAT

Pokud chceme navrhnout pokojový termostat, musíme si vyjmenovat všechny požadavky, které na něj budeme klást. V práci [17] byla vypracována podrobná rešerše pokojových termostatů, ze které vybereme takové požadavky, aby námi navrhovaný termostat měl nejlepší vlastnosti a funkce. Termostaty řízené mikroprocesorem spadají podle [17] do kategorie elektronické programovatelné.

Hlavními požadavky v této kategorii je kvalitní regulace teploty, dále s tímto požadavkem souvisí i úspora energií, potom komfortní ovládání termostatu, možnost nastavení různých teplot nebo volba přednastaveného programu, zobrazení informací o regulovaném procesu, v neposlední řadě cena a design.

Mezi vedlejší požadavek můžeme zařadit bezdrátový přenos akčního zásahu, který výrazně zvyšuje cenu termostatu. Na druhou stranu je možné termostat použít v kterékoliv místnosti, což může výrazně přispět k tepelné pohodě v domě.

V našem návrhu se omezíme na termostat pevně zabudovaný v referenční místnosti. Budeme tedy navrhovat termostat podle hlavních požadavků.

2.1.1 Kvalita regulace

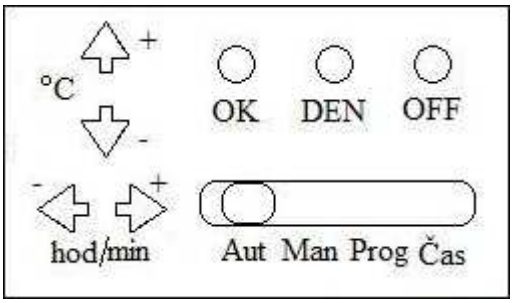
Abychom dosáhli kvalitní regulaci, je nutné nastavit optimální statické a dynamické vlastnosti systému. Této problematice se budeme věnovat podrobněji v kapitole 5 při návrhu algoritmů řízení. Pro návrh termostatu je důležité, že akční zásah v systému je omezen pouze na zapnutí a vypnutí zdroje (kotle). Jedná se tedy o nespojitý akční zásah. Z tohoto vyplývá nutnost použití některého spínacího výkonového prvku, kterým budeme zapínat nebo vypínat zdroj tepla. Je tedy zřejmé, že regulovaná veličina bude kmitat kolem požadované v pásmu nastavené hystereze. Dalším důležitým faktorem pro návrh termostatu je přesnost měření teploty. Při regulaci teploty v místnosti je dostačující přesnost nastavení a měření teploty ± 0,5 °C.

2.1.2 Komfortní ovládání

U většiny termostatů je ovládání složité. Používání jednoho tlačítka pro více voleb může být pro někoho velký problém. Na druhou stranu velký počet tlačítek zhoršuje orientaci při ovládání. Možné uspořádání ovládacích prvků pokojového termostatu je vyznačeno na obr. 1.

Námi navržený termostat by měl mít tyto ovládací prvky:

1. Dvě tlačítka pro komfortní nastavování teploty po kroku 0,5 °C
2. Dvě tlačítka pro nastavení času
3. Přepínací tlačítko pro možnosti volby mezi režimy (Aut, Man, Prog, Čas) termostatu
4. Další tlačítka pro potvrzení volby (OK), nastavení dne v týdnu (DEN) a vypnutí regulace (OFF)
5. Případně, další tlačítka pro některé další funkce



Obr. 1: Ovládací panel navrhovaného termostatu

2.1.3 Funkce

Použití mikroprocesoru, s sebou nese značné výhody v programování funkcí termostatu. Samozřejmostí každého termostatu je možnost nastavit si různé teploty v různý časový okamžik. Dále to může být funkce prázdninový provoz, možnost dálkového ovládání, kalibrace čidla, zamknutí displeje, atd.

2.1.4 Zobrazování informací

Většina dnešních termostatů mívá kvalitní zobrazení informací o regulovaném procesu, ale v některých případech není rozvržení informací na displeji ideální. Tímto máme na mysli, např. nesmyslně velké zobrazení času a dalších nepotřebných informací. Termostat je především regulátor teploty a na to někteří výrobci zapomínají. Zobrazování na navrhovaném termostatu bude řešeno alfanumerickým maticovým LCD displejem. Podrobný popis displeje je v kapitole 3.2.1

Navržený termostat by měl mít na displeji zobrazeny tyto informace:

1. Aktuální teplota
2. Požadovaná teplota
3. Čas
4. Indikaci dne v týdnu a zapnutí kotle
5. Případně, další zobrazení jako zvolený režim, slabá baterie, číslo programu

2.1.5 Cena

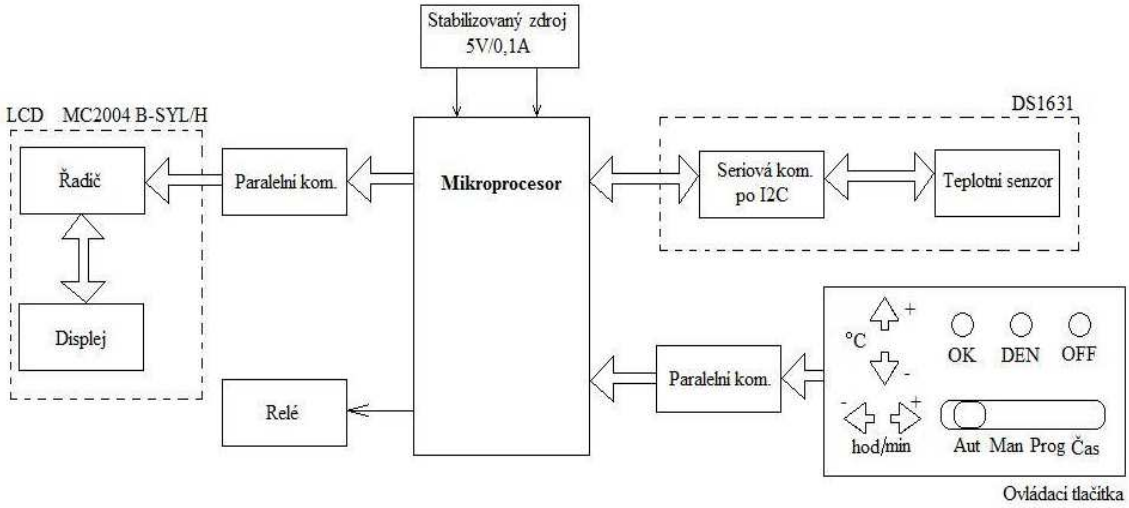
Cena elektronických programovatelných termostatů se pohybuje v rozmezí od 700 - 3000 Kč. Přičemž největší položkou je displej. Dále pak počet programové vybavení termostatu.

2.1.6 Shrnutí

- Přesné sledování žádané teploty a kompenzace poruchy
  - ON/OFF – volba hystereze
  - Pokročilý algoritmus řízení
- Zajištění ne příliš častého spínání zdroje tepla
- Rozlišovací schopnost snímače teploty 0,1 °C
- Jednoduchá obsluha - dostatečný počet ovládacích prvků
- Snadno čitelné a přehledné zobrazování zejména žádané a aktuální teploty v místnosti, režimu termostatu a stavu zdroje tepla,
- Nízká cena

3. HARDWAROVÝ NÁVRH

Návrh termostatu je založen na blokovém schématu z obr. 2. Blokové schéma bylo navrženo podle výše uvedených požadavků.



Obr. 2: Blokové schéma termostatu

3.1 VSTUPNÍ PERIFERIE

Vstupními periferiemi máme na mysli čidlo teploty a ovládací tlačítka. V dalším si podrobněji popíšeme jednotlivé periferie a uvedeme konkrétní případy periferií použitých v návrhu termostatu.

3.1.1 Čidlo teploty

Úkolem čidla je zprostředkovat informaci o stavu měřené veličiny. Abychom vybrali správné teplotní čidlo pro náš termostat, je nutné udělat krátký úvod k problematice měření teploty. Snímat teplotu můžeme několika základními způsoby. Zde je uvedeno rozdělení podle [16].

Dělení senzorů podle fyzikálního principu:

- Elektrické (odporové kovové, odporové polovodičové, polovodičové s PN přechodem, termoelektrické, krystalové)
- Dilatační (kapalinové, plynové, parní a bimetalové)
- Speciální (akustické, šumové, magnetické, tekuté krystaly, aj.)

Dělení senzorů podle vzájemného působení:

- Dotykové
- Bezdotykové

Dělení senzorů podle typu výstupního signálu:

- Analogové
- Digitální

Odporové kovové senzory teploty [16] [9]

Principem těchto senzorů je teplotní závislost odporu kovu na teplotě. Rozsah, přesnost, linearita, stálost a někdy i konstrukce čidla určuje materiál použitý pro výrobu čidla. Základní materiálovou konstantou je teplotní součinitel odporu α_R, dalšími důležitými parametry materiálu je rezistivita, čistota a změna teplotního součinitele odporu s časem.

Pro provozní kovové teploměry lze v rozsahu teplot 0 °C až 100 °C použít vztah:

$$R_t = R_0(1 - \alpha_R(\lambda_t - \lambda_0)) \qquad [\Omega] \qquad (1)$$

kde R_0 [Ω]je odpor při teplotě 0 °C, R_t [Ω]je odpor při teplotě t a α_R [$°C^{-1}$] je teplotní součinitel odporu.

Nejrozšířenějšími odporovými kovovými teploměry jsou teploměry platinové. Platina se používá pro svou dobrou chemickou netečnost, časovou stálost a vysokou teplotu tání. V tabulce Tab. 1 je přehled dalších kovů použitelných pro odporové kovové teploměry.

Materiál	Měrný odpor [Ωm]	α_R [$°C^{-1}$]	Rozsah použití [°C]
Platina	$9{,}81 \cdot 10^{-6}$	$3{,}85 \cdot 10^{-3}$	-250 ÷ 850 (1200)
Nikl	$9{,}13 \cdot 10^{-6}$	$6{,}17 \cdot 10^{-3}$	-60 ÷ 150
Měď	$1{,}66 \cdot 10^{-6}$	$4{,}27 - 4{,}33 \cdot 10^{-3}$	do 100 ÷ 150
Molybden	$1{,}66 \cdot 10^{-6}$	$5{,}5 \cdot 10^{-6}$	–

Tab. 1: Přehled nejpoužívanějších kovových teploměrů [9]

Odporové polovodičové senzory teploty [16] [9]

Princip těchto snímačů je podobný jako u odporových kovových, avšak zde se využívá teplotní závislost polovodiče na teplotě. Polovodičové odporové senzory dělíme na:

- Termistory -Negastory
  -Pozistory
- Monokrystalické senzory

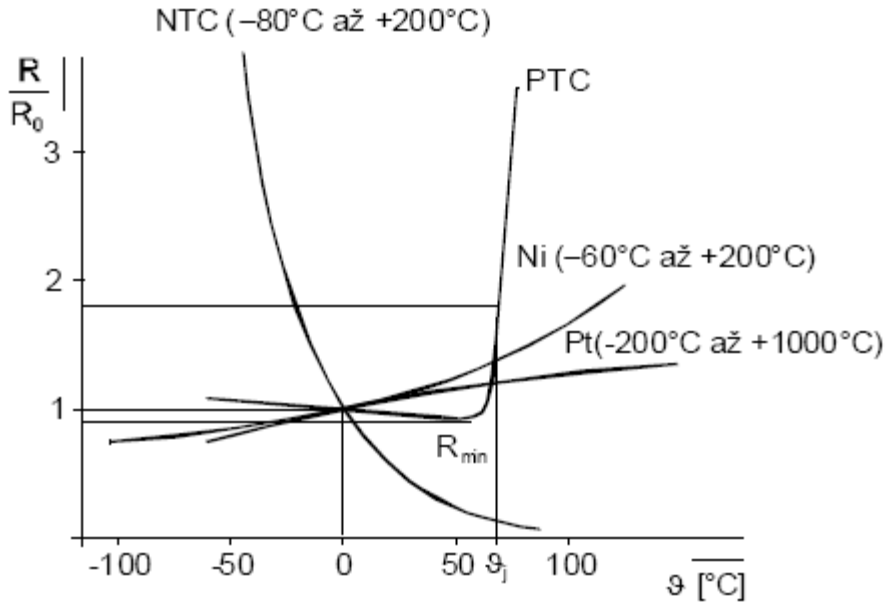
Negastory (NTC) mají záporný teplotní součinitel odporu. S rostoucí teplotou jejich odpor klesá. Teplotní součinitel odporu je 5-50krát větší než u kovů. Vyrábějí se práškovou metalurgií. Používají se především pro nízkoteplotní termistory v rozsahu 4,2 až 600K. Maximální teplotní rozsah je do 1000 °C. Závislost odporu NTC termistoru na teplotě je dána vztahem

$$R = R_0 e^{-B(\frac{1}{T_0}-\frac{1}{T})} \quad [\Omega] \qquad (2)$$

kde R_0 [Ω]je odpor NTC termistoru při teplotě T_0 [K], R [Ω] je odpor NTC termistoru při teplotě T [K] a B je materiálový součinitel. Hlavní výhodou jsou malé rozměry, vysoký teplotní součinitel odporu a vysoká citlivost. Základní nevýhodou je značná nelinearita.

Pozistory (PTC) mají kladný teplotní součinitel odporu. S rostoucí teplotou dochází k mírnému poklesu odporu a v určitém úzkém rozsahu teplot odpor prudce stoupá a dále opět mírně klesá. Změna odporu o 50% odpovídá tzv. spínací teplotě. Tvaru charakteristiky pozistoru lze využít k indikaci určité teploty.

Rozdíly tvaru charakteristik různých odporových čidel teploty jsou patrné z obr. 3.



Obr. 3: Teplotní závislost odporových snímačů teploty [9]

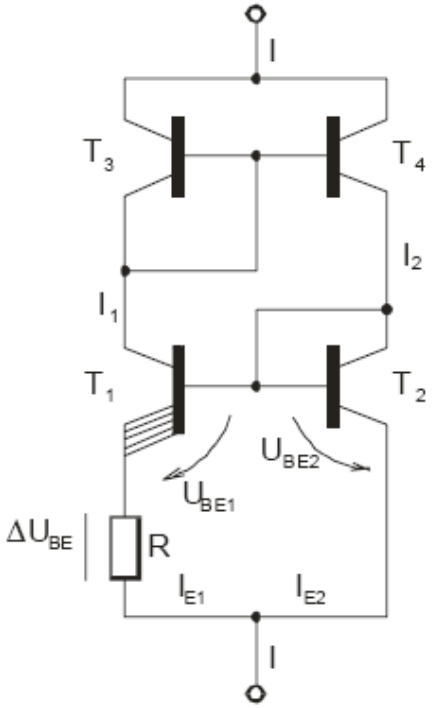
Monokrystalické senzory s PN přechodem [16] [9]

PN senzory (diody, tranzistory) teploty jsou založeny na teplotní závislosti napětí PN přechodu v propustném směru. Používají se v teplotním rozsahu 1 až 400K. Pro realizaci se používá především křemík, germanium a galium-arzenid. Výhodnou vlastností polovodičové diody (tranzistoru v diodovém zapojení) jako čidla teploty je lineární závislost výstupní termometrické veličiny (úbytku napětí na přechodu) na teplotě. Při napájení diody konstantním proudem je úbytek napětí na PN přechodu dán vztahem:

$$U_D = \frac{nk}{e}\ln(1+\frac{I}{I_S})T \qquad [V] \qquad (3)$$

kde k je Boltzmannova konstanta, n je činitel polovodiče, e je elementární náboj a I_S je nasycený proud diodou ve zpětném směru. Diodová čidla se vyznačují velkou citlivostí, linearitou a miniaturními rozměry. Díky miniaturním rozměrům je možné tyto senzory integrovat do tzv. integrovaných PN snímačů teploty obr. 4. Tento snímač je napájen ze zdroje napětí U a výstupem je velikost výstupního proudu I úměrný měřené teplotě. V některých případech se do pouzdra teplotního snímače může zabudovat číslicový modulátor, který plní funkci A/D převodníku.

V takovýchto případech nemá výstupní veličina analogový charakter, ale je převedena na binární číslo.



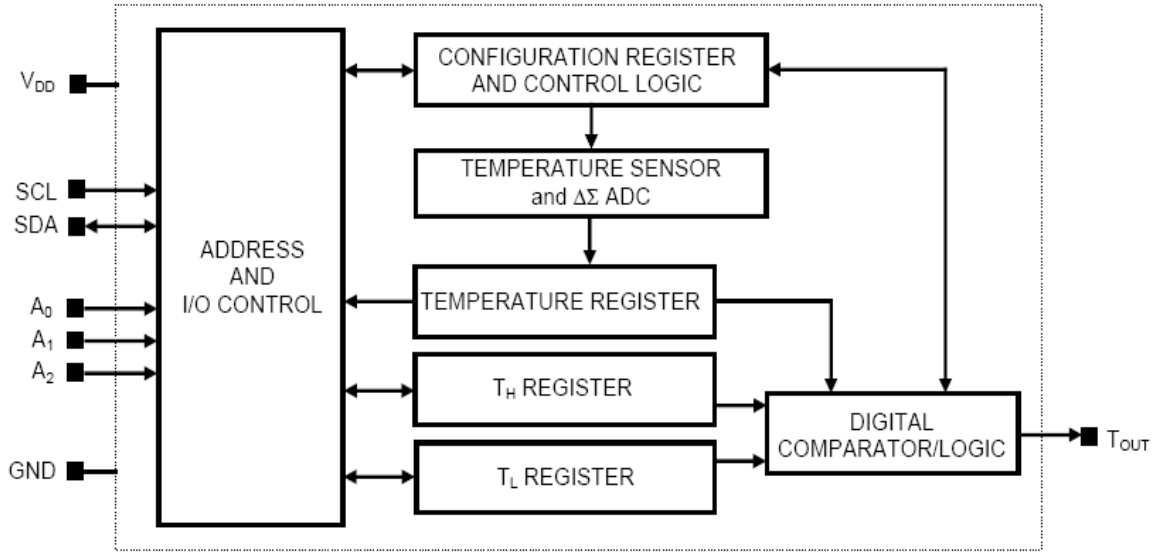
Obr. 4: Integrovaný PN snímač teploty [9]

DS1631 [1]

Pro náš návrh jsme vybrali teplotní čidlo DS1631. Toto čidlo můžeme podle výše zmíněného zařadit mezi integrované teplotní senzory dotykové, polovodičové s PN přechodem a s digitálním výstupem. Pro naše účely je toto čidlo vyhovující, protože nároky na přesnost měření nejsou tak vysoké a použití čidla s digitálním výstupem nám umožní okamžitě zpracovávat získané vzorky mikroprocesorem. Bez nutnosti použití externího A/D převodníku.

Čidlo vyrábí firma Dallas Semiconductor, jedná se o inteligentní senzor teploty s 9, 10, 11 nebo 12 bitovým datovým výstupem a se sériovou komunikací I^2C (MSB první). Na obr. 5 je znázorněno blokové schéma senzoru. Přenáší se vždy 2 bajty (MSB a LSB). Druhý z přenášených bajtů (LSB) udává rozlišení (maximální rozlišení 0,0625 °C), podle toho kolika bitový si nastavíme přenos. Typ rozlišení se

nastavuje v konfiguračním registru (obr. 6) bity R0 a R1. Identifikace senzoru na sběrnici je 7bitovou adresou, jejíž nejnižší 3 bity je možné měnit uzemněním externích vývodů (A0, A1, A2). Měření teploty je prováděno pomocí integrovaného PN teplotního senzoru, který je v zapojení typu bandgap. Analogová informace o teplotě je převáděna pomocí delta-sigma A/D převodníku. Měření teploty může být prováděno kontinuálně (continuous-conversion mode) nebo jednorázově na příkaz (one-shot mode). Dále je možné využít funkci termostatu, kdy se na vývodu TOUT objevuje stav log. 1 při překročení hodnoty uvedené ve vnitřním registru TH a naopak stav log. 0 při hodnotě teploty pod hodnotou v registru TL. Rozdílné hodnoty v obou registrech tak definují hysterezi překlápění výstupu.



Obr. 5: Blokové schéma DS1631 [1]

MSb	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	LSb
DONE	THF	TLF	NVB	R1	R0	POL*	1SHOT*

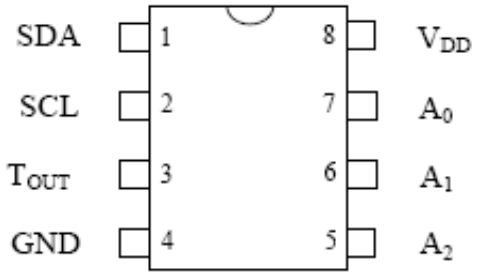
*NV (EEPROM)

Obr. 6: Konfigurační registr DS1631 [1]

Charakteristické rysy:

- Měřící rozsah od -55 °C do 125 °C s přesností ±0,5 °C
- Výstupem je 9, 10, 11 nebo 12 bitové číslo (2 přenesené bajty)
- Široký rozsah napájecího napětí (2.7 až 5.5 V)
- Převod teploty trvá max. 750ms
- Data jsou čtena pomoci sériového rozhraní I^2C
- 8 pin DIP pouzdro (obr. 7)
- Spotřeba: 1 mA (800nA v standby modu-záložním módu)

Popis vývodů:



SDA	- (I^2C) datová linka
SCL	- (I^2C) hodinový vstup
T_{OUT}	- výstupní signál z termostatu
GND	- zem
V_{DD}	- napájení
A0-A3	- vstupy pro nastavení adresy

Obr. 7: Pouzdro 8-PIN DIP čidla DS1631 [1]

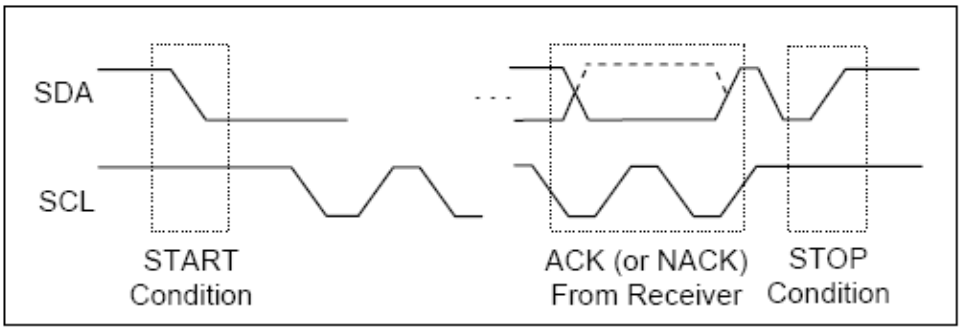
Popis I^2C komunikace u čidla DS1631 [1]

Jak již bylo řečeno čidlo DS1631 komunikuje s mikroprocesorem pomocí univerzální sériové sběrnice I^2C. Sběrnice je určena pro komunikaci typu master/slave. Čidlo můžeme rozlišit pomocí 7 bitové adresy zasílané v tzv. řidícím bajtu (obr. 8), při zahájení každé komunikace. Nejméně významný 8 bit určuje směr přenosu dat, kde 0 = zápis do obvodu a 1 = čtení z obvodu. Řidící vodiče jsou tedy dva, SDA (data) a SCL (hodiny). Oba řidící signály se připojují přes pullup rezistory na napájení, aby vodiče zůstaly při nečinnosti v log. 1.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	1	A_2	A_1	A_0	$R/\overline{W}$

Obr. 8: Řidící bajt čidla DS1631 [1]

Komunikace se zahajuje tzv. START podmínkou, kdy signál SCL zůstává v log. 1 a signál SDA se stáhne k nule. Ukončení komunikace je tzv. STOP podmínkou, kdy se signál SDA nastaví na log. 1 při signálu SCL v log. 1. Data se přenášejí od nejvýznamnějšího bitu k nejméně významnému. Stav signálu SDA se může měnit, pouze pokud je signál SCL v nule. Nastavení signálu SCL oznamuje obvodu platnost dat. Při čtení ze sběrnice se signál SDA nastaví na +5V a data se načítají při nastaveném signálu SCL. Na konci přenášení každého bajtu se ještě přenáší 9. kontrolní bit ACK, jehož vynulováním obvod provede potvrzení příjmu. Při příjmu dat nuluje tento bit řidící obvod master a to pouze má li komunikace pokračovat. Pokud bude po příjmu následovat podmínka stop tak se tento poslední bit nastaví do log. 1. Různé druhy stavů vyskytující se u I^2C komunikace jsou znázorněny na obrázku 9.



Obr. 9: Stavy u I^2C komunikace [1]

Příkazy potřebné ke změření teploty:

Start Convert T [51h] – začne měřit teplotu a převádět na binární číslo

Stop Convert T [22h] – zastaví měření a převod při režimu (continuous conversion mode)

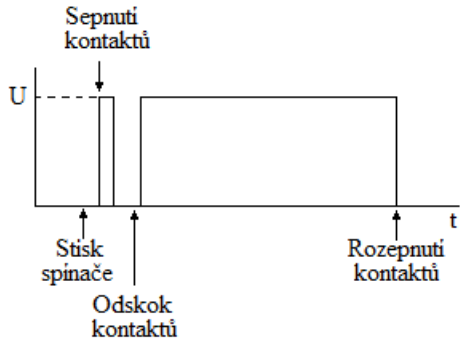
Read Temperature [AAh] – přečte poslední změřenou a převedenou teplotu z 2-byte teplotního registru

3.1.2 Ovládací tlačítka

Dalším prvkem vstupních periferií jsou ovládací tlačítka. Ovládací tlačítka plní u termostatu nastavovací funkci. V další části si popíšeme některé problémy, vznikající při používání tlačítek a způsoby jejich připojování.

Ošetření zakmitávání kontaktů [11]

Tlačítka, jejichž kontakty jsou skokem spínány a rozepínány, vykazují při sepnutí odskoky kontaktů. Tato situace je znázorněna na obr. 10. Podobné průběhy mohou vzniknout i při příchodu dostatečně velkých rušivých impulzů na vstup mikroprocesoru.



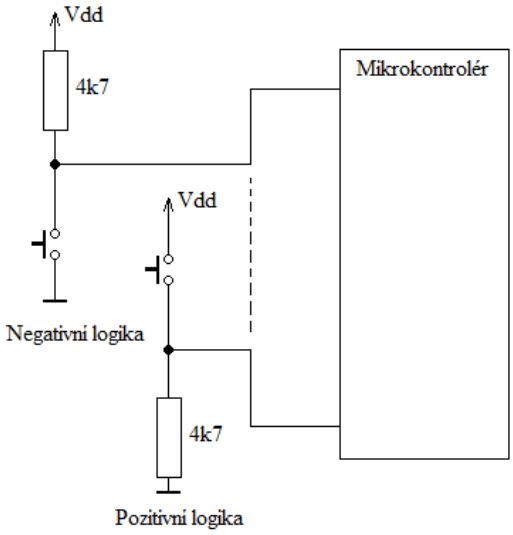
Obr. 10: Vznik odskoků na spínači [11]

Tento jev lze odstranit dvěma způsoby. **Softwarově** - způsobem snímání stavů vstupů nebo **hardwarově** - použitím obvodů, které ošetřují kmitání tlačítek, použitím tlačítek, která nezakmitávají, použití filtrace krátkých pulzů.

Způsoby připojení tlačítek

Přímé připojení ke vstupům [11]

Přímé připojení se využívá v případě, že máme k dispozici dostatečný počet vstupů mikroprocesoru, které můžeme pro tento účel použít. V našem případě je nutné připojit pouze 11 tlačítek. Je tedy zřejmé, že použijeme přímé připojení. Na obr. 11 jsou nakresleny dva způsoby přímého připojení tlačítka k mikroprocesoru. U přímého připojení se využívá dvou způsobů tzv. negativní a pozitivní logika. Ve většině případů se však používá negativní logika, protože u toho způsobu dokážeme zjistit poruchu přerušení vodičů. Princip negativní logiky spočívá v tom, že vstup je přes pullup rezistor připojen na napájení. Vstup je tedy při rozepnutí tlačítka v log. 1 s při stisknutí je na vstupu log. 0.



Obr. 11: Přímé připojení tlačítka k mikrokontroléru

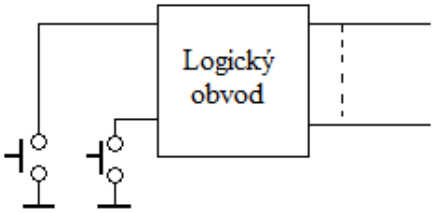
Klávesnice [11]

Někdy potřebujeme připojit více tlačítek, než máme k dispozici vstupů. Toho lze dosáhnout například vytvořením klávesnice, která bude generovat kód nebo se bude jednat o multiplexně snímanou klávesnici.

U klávesnic generujících kód jsou tlačítka připojena na vstupy kombinačního logického obvodu, který v závislosti na stisknutém tlačítku generuje kód (obr. 12).

Logický obvod může být realizován jednoduše pomocí diodové matice nebo programovatelným logickým obvodem.

Další ze způsobů připojení klávesnice je velmi často využívané zapojení do matice nebo použití multiplexoru. Multiplexor pracuje tak, že na výstupu se nachází negovaný stav vstupu D0/15, který je adresován vstupy S3/0.



Obr. 12: Blokové schéma klávesnice generující kód [11]

3.2 VÝSTUPNÍ PERIFERIE

Výstupními periferiemi máme na mysli LCD displej a výkonový prvek, který bude vykonávat akční zásah, protože termostat je především regulátor teploty. V další části si podrobněji popíšeme jednotlivé periferie a uvedeme konkrétní případy použité v našem návrhu.

3.2.1 LCD Displej

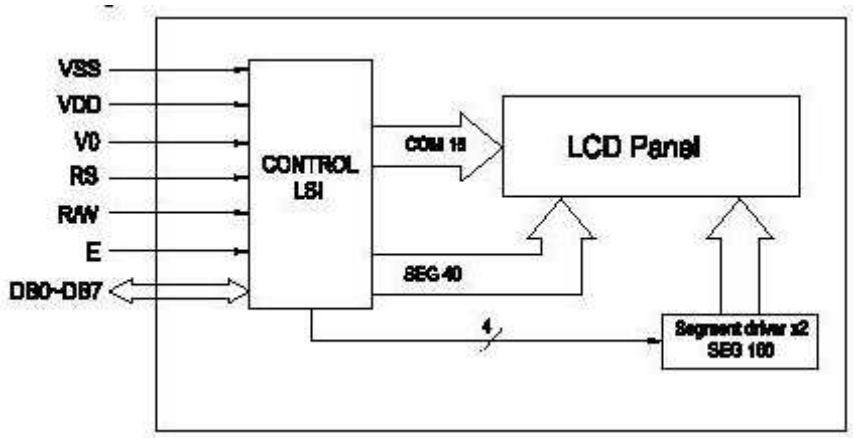
Displej u termostatu plní informativní funkci. Měl by zobrazovat informace o aktuální teplotě, o nastavené teplotě, provozu kotle a mnoho dalších podle programového vybavení termostatu.

Kritéria podle, kterých můžeme vybírat LCD displej jsou:

- Struktura segmentů (segmentová, alfanumerická, grafická)
- Rozměr (8x2, 16x2, 20x4 a další)
- Komunikace s mikroprocesorem (s řadičem, bez řadiče)
- Cena a design

Jak již bylo řečeno v kapitole 2.1.4, displej by měl zobrazovat většinu informací o regulovaném procesu, a proto není možné volit segmentový LCD displej, který je sice velmi levný, ale nebylo by možné na něm zobrazit požadované informace. Naproti tomu grafická struktura segmentů displeje by rozhodně vyhovovala požadavkům na zobrazování informací, ovšem takový displej je dražší než odpovídající alfanumerický displej, na kterém lze zobrazit stejné informace.

Z těchto důvodů byl vybrán alfanumerický maticový LCD displej MC2004B-SYL/H, který má rozměr 20x4, kde první číslo uvádí počet znaků na řádek a druhé číslo uvádí počet řádků. Displej má transreflexní žlutozelené podsvětlení. Komunikaci s mikroprocesorem zajišťuje řadič, který je integrován na DPS společně s displejem. Jedná se o standardní řadič HD44780 od firmy Hitachi. Tento řadič nebo nějaký jeho ekvivalent používají všichni výrobci displejů dále je zapojení přívodního konektoru stejné, čímž je zaručena úplná kompatibilita. Na obr. 13 je znázorněno blokové schéma displeje.



Obr. 13: Blokové schéma displeje [4]

Displej může komunikovat s mikroprocesorem po čtyřech (DB4-DB7) nebo osmi (DB0-DB7) datových linkách. Pracuje-li po čtyřech datových linkách, posílají se do něj data dvěma zápisy. Nejdříve horní, a pak dolní půlbajt. [11]

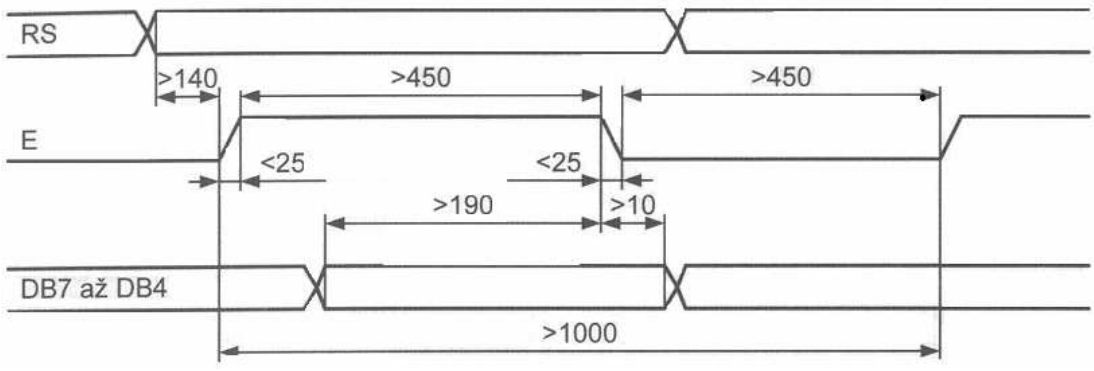
Prostřednictví signálů RS, R/-W a E se nastaví komunikace s mikroprocesorem. Význam signálů je uveden v tab. 2. Pomocí těchto signálů a datových vodičů jsou do LCD displeje přenášeny příkazy a data určená k zobrazení,

popřípadě čtena data CG RAM (paměti generátoru znaků), DD RAM (paměti zobrazovaných dat), znaky busy a adresa čítače kurzoru. [11]

Číslo pinu	Signál	Popis
1	V_{SS}	Zem (0V)
2	V_{DD}	Napájecí napětí (4,75-5,25)
3	V_0	Nastavení kontrastu (min. 0,65 V)
4	RS	Výběr registru
5	R/-W	Volba čtení nebo zápisu
6	E	Vstup povolení
7-14	DB0-DB7	Data/kód 0 -Data/kód 7
15	V_{LED}	Napájení podsvícení
16	V_{LSS}	0 V

Tab. 2: Obsazení vývodů displeje [11]

Časové průběhy přenosu dat nebo příkazů mezi mikroprocesorem a displejem jsou na obr. 14. Časové relace pro ovládání displeje jsou velmi rychlé, proto není těžké dodržet časové posloupnosti bez vkládání čekacích smyček. Stačí čtyři kroky pro provedení zápisu nebo čtení do/z displeje. [11]



Obr. 14: Časové průběhy zápisu dat do displeje (uvedené časy jsou v ns) [11]

Po zapnutí napájení je hardwarově proveden reset displeje. Před zahájením komunikace je nutné provést inicializaci displeje. K tomu slouží posloupnost přesně

definovaných příkazů, které musíme zapsat do displeje. Podrobný popis příkazu najdete v [11].

3.2.2 Spínací výkonový člen

Jak již bylo řečeno v kapitole 2.1.1 akční zásah, termostatu bude převáděn na dvoustavový ON/OFF. Musíme tedy zajistit spínání zdroje tepla. Spínání můžeme realizovat pomocí relé, silových polovodičových spínačů nebo galvanicky oddělenými součástkami (optrony). Pro náš návrh jsme zvolili klasické relé, kvůli jeho dobrým vlastnostem a nízké ceně.

Výhody relé:

- Nulový úbytek na spínacím kontaktu
- Galvanické oddělení
- Velká přetížitelnost
- Malé rozměry
- Malý ovládací příkon
- Nízká cena

Nevýhody relé:

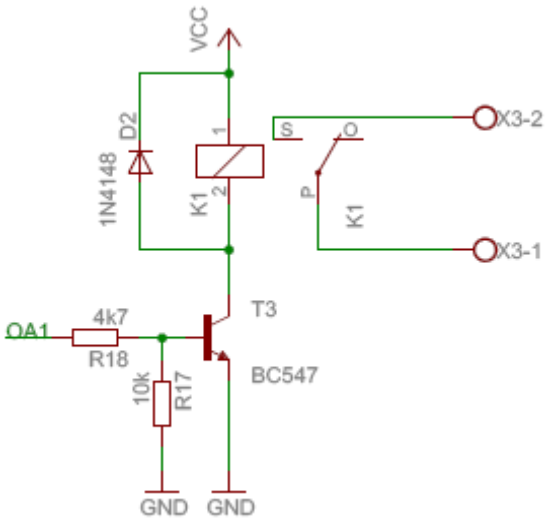
- Opalování spínacích kontaktů

S použitím moderních technologií se úspěšně daří odstraňovat jedinou nevýhodu relé. Doba životnosti a spolehlivost relé se neustále zvětšuje. Je tedy zřejmé, že relé je často používaným konstrukčním dílem.

Parametry podle, kterých můžeme vybírat relé: [14]

- Charakter zátěže (odporová, induktivní, kapacitní)
- Hodnota zátěže: velký proud (výkonová, automobilová) malý proud (signálová a VF)
- Požadavky na životnost

- Napájení cívky (střídavé nebo stejnosměrné)
- Napětí cívky a její odpor (spotřeba)
- Spínací a rozpínací napětí



Obr. 15: Schéma zapojení relé

V našem návrhu jsme vybrali relé RELEF4161-12. Jedná se o relé s DC cívkou. Spínací napětí je 12V. Trvalý spínaný proud 16A. Odpor relé 360Ω. Maximální spínaný výkon 4000VA. Na obr. 15 je znázorněno schéma zapojení relé. Ovládáním výstupu mikroprocesoru spínáme tranzistor T3, který následně sepne relé. Dioda odvádí při rozpojení tranzistoru energii akumulovanou v indukčnosti zátěže a tím brání vzniku vysokého indukovaného napětí.

3.3 MIKROPROCESOR

Při výběru mikroprocesoru je nutné uvažovat všechny předchozí součástky zmíněné v návrhu. Je tedy nutné, aby vybraný mikroprocesor měl odpovídající počet vstupů a výstupů. V našem případě musí mít minimálně 8 paralelních výstupů pro ovládání LCD displeje, 1 paralelní výstup pro výkonový akční člen a 11 paralelních vstupů pro tlačítka. Dále by měl, disponovat potřebnými komunikačními sběrnicemi a rozhraním. Vybraný mikroprocesor musí obsahovat rozhraní I^2C pro komunikaci

s čidlem teploty, SPI rozhraní pro programování v aplikaci. Dalším kritériem pro výběr mikroprocesoru může být počet přerušení, časovačů/čítačů a případně některé další komunikační rozhraní (např. USART). V neposlední řadě je nutné vybrat velikost pamětí a typ procesoru (8bit, 32bit). Tyto hodnoty jsou přímo závislé na počtu funkcí naprogramovaných v termostatu a složitostí výpočtů při nich vykonávaných.

Pro náš termostat jsme podle předchozích požadavků a dřívějších zkušeností s mikroprocesory AVR od firmy ATMEL, vybrali mikroprocesor AVR ATmega64-16AU. Jedná se o nízko příkonový 8bitový mikroprocesor z výrobní řady ATmega. Procesor je založen na RISC architektuře. Procesor je uložen v smd pouzdru TQFP64. Disponuje 53 programovatelnými vstupy/výstupy.

Vlastnosti ATmega64-16AU: [6]

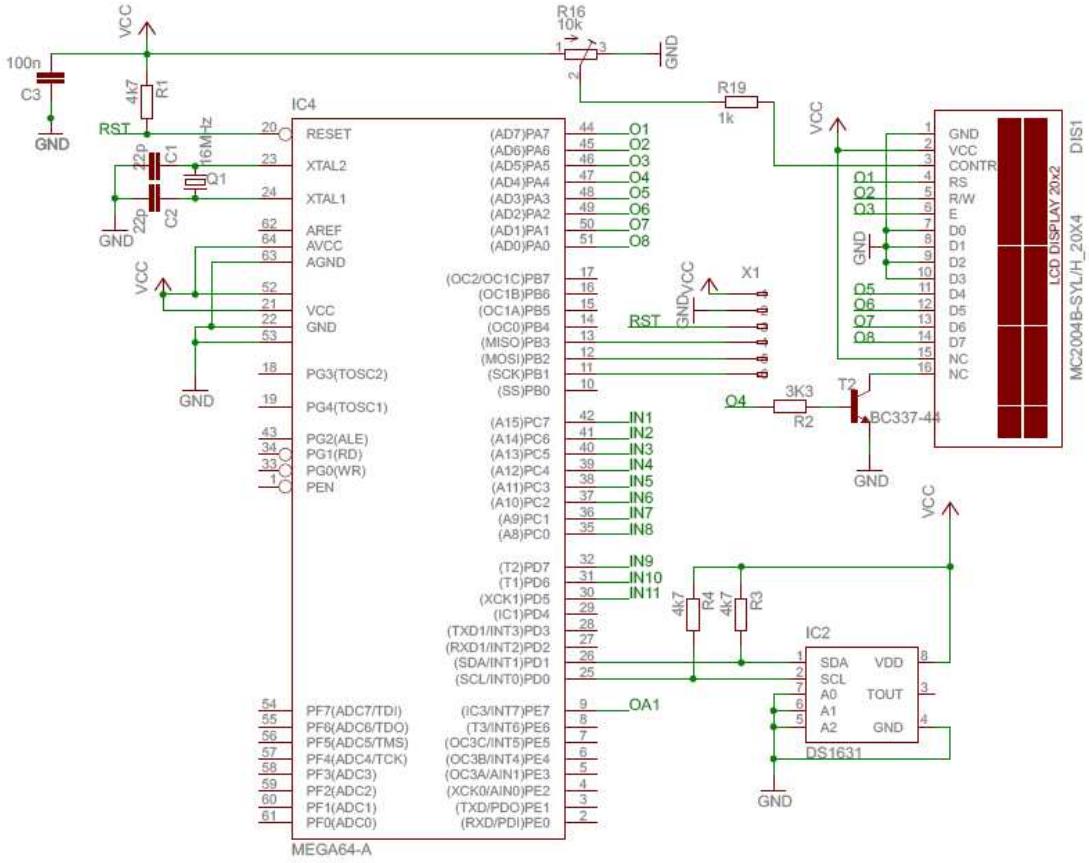
- 64kB program FLASH, 4kB SRAM, 2kB EEPROM
- 2x8bit, 2x16bit časovačů/čítačů,
- 6+2 PWM kanálů
- 8x10bit ADC
- I2C, 2xUSART, SPI
- Vnitřní RC oscilátor
- Watch-dog, úsporné režimy

Mikroprocesory AVR je možné programovat ve vývojovém prostředí AVR Studio® volně dostupné na stránkách výrobce. Prostředí podporuje psaní kódu v obou nejrozšířenějších způsobech a to psaní v jazyce C i v assembleru. Podrobnější informace o mikroprocesoru jsou uvedeny v literatuře [6].

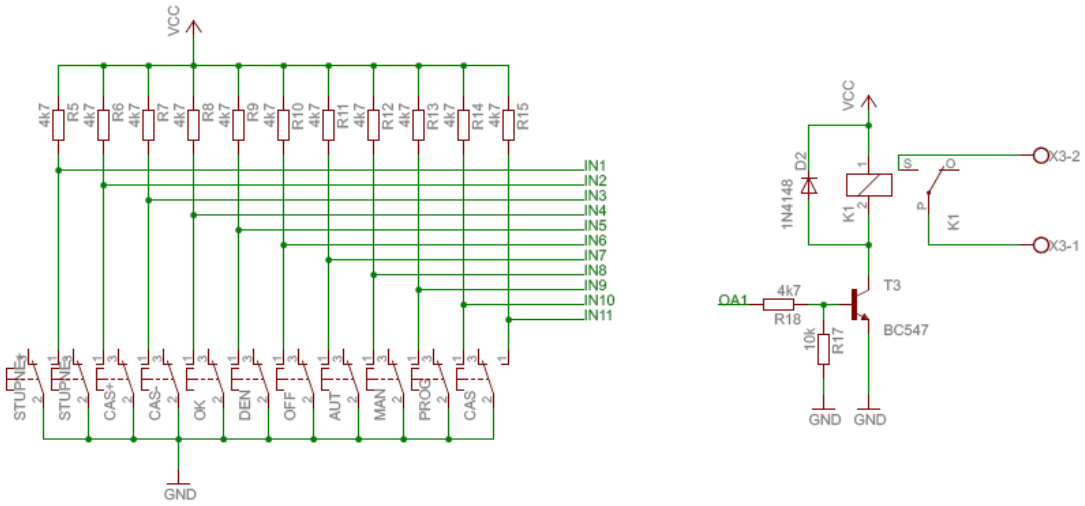
3.4 NAPÁJENÍ

Termostat je napájen ze stabilizovaného napětí 5V. Použili jsme nízko přepěťový stabilizátor L4940V05, který dává proud 1A. Odběr proudu celého obvodu je v řádech desítek mA, a proto je tato hodnota více než dostačující.

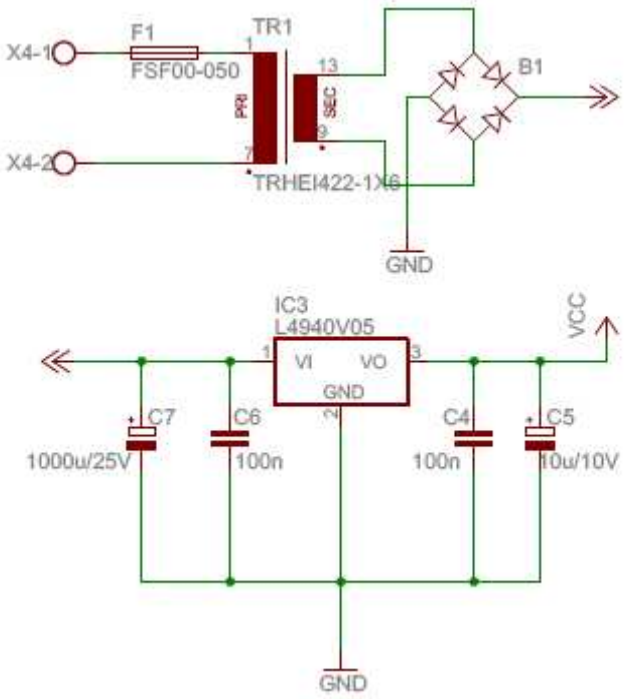
4. ELEKTRICKÉ SCHÉMA NAVRŽENÉHO TERMOSTATU



Obr. 16: Elektrické schéma navrženého termostatu část 1



Obr. 17: Elektrické schéma navrženého termostatu část 2



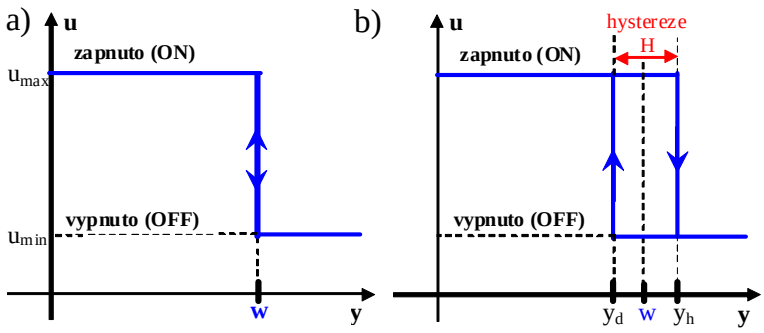
Obr. 18: Elektrické schéma navrženého termostatu část 3

5. NÁVRH ALGORITMŮ REGULACE TEPLOTY

V této části práce budou podrobně popsány námi navržené algoritmy regulace teploty. V práci [17] byl navržen měřením a statistickým vyhodnocením této analýzy verifikován matematický model místnosti v programu Matlab-Simulink. Pomocí tohoto modelu jsme nastavili nejlepší parametry algoritmů, tak aby byla regulace vyhovující i v reálném objektu.

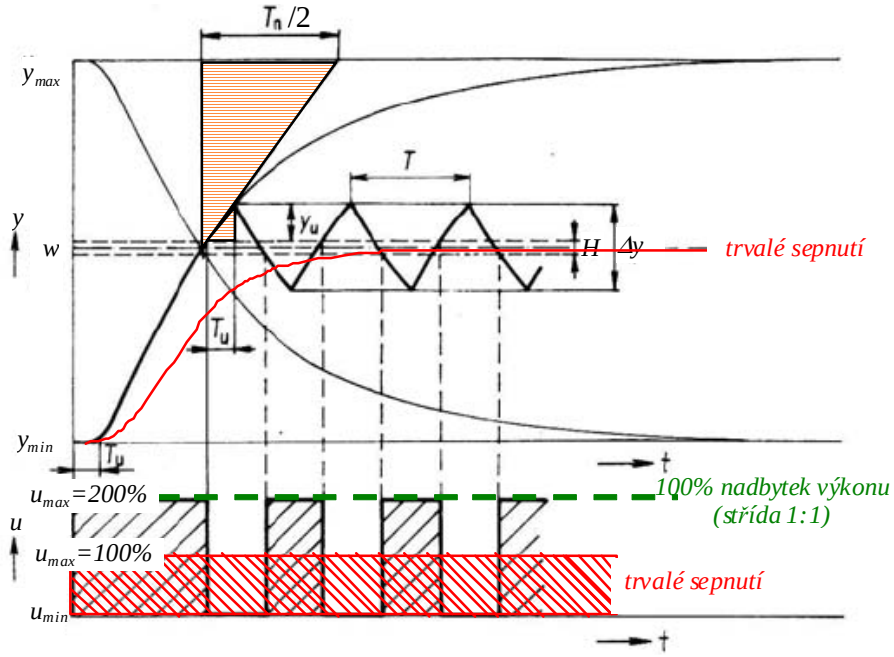
5.1 ON/OFF REGULÁTOR [17]

Jedná se o dvoupolohový regulátor. Jestliže poklesne skutečná hodnota regulované veličiny pod žádanou hodnotu, přepne se regulátor do stavu ON (zapnuto). Akční veličina tedy nabude nějaké pevné hodnoty u_{max}. Překročí-li naopak skutečná hodnota regulované veličiny žádanou hodnotu, přepne se regulátor do stavu OFF (vypnuto). Akční veličina nabude jiné pevné hodnoty u_{min} (obr. 19a).



Obr. 19: Statická převodní charakteristika ON/OFF regulátoru a) bez hystereze, b) s hysterezí [17]

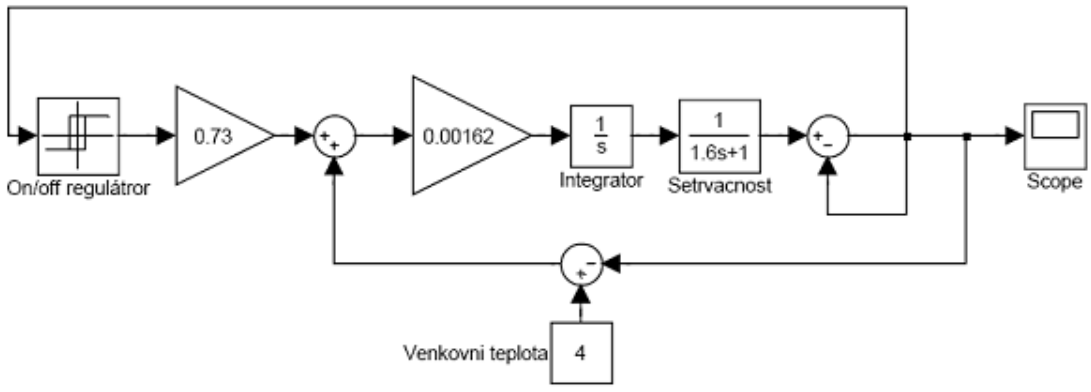
Aby při malých oscilacích nedocházelo k opalování kontaktů relé, zavádí se do statické charakteristiky ON/OFF regulátoru hystereze (obr. 19b). Regulovaná veličina tedy kmitá v pásmu hystereze se šířkou H. Volba hystereze je vždy otázkou kompromisního řešení. U statických soustav 2 a vyššího řádu již kmitání regulované veličiny nezávisí pouze na hysterezi, ale i na vlastnostech samotné soustavy. Šířka pásma kmitání regulované veličiny Δy a tím i kvalita regulačního pochodu, je závislá především na době průtahu T_u soustavy (obr. 20).



Obr. 20: Průběh regulované veličiny y a akční veličiny x při regulaci statické soustavy 2. řádu ON/OFF regulátorem s hysterezí [17]

Zlepšení kvality regulačního pochodu lze těmito způsoby:

- Zmenšení hystereze
- Zkrácení doby průtahu
- Prodloužení doby náběhu
- Zmenšení rozsahu akční veličiny



Obr. 21: Simulační schéma pro ON/OFF regulátor

5.1.1 Návrh regulátoru

Na obr. 21 je znázorněno simulační schéma ON/OFF regulátoru s modelem místnosti. Jeho spínací a rozpínací hodnoty jsou nastaveny na maximální a minimální

hodnoty teplot, které určuje hystereze. Hystereze regulátoru je 0.4. Při realizaci ON/OFF regulátoru by bylo dobré umožnit uživateli volbu hystereze. Vzhledem k tomu, že každá otopná soustava má různé časové konstanty může regulovaná veličina kmitat v rozmezí vyšším než je nastavená šířka hystereze. Tato volba by mohla přispět ke zlepšení regulace. Ovšem volbu hystereze je nutné vždy náležitě zvážit, protože malá hystereze vede k příliš častému spínání zdroje tepla a to může vést ke snížení jeho účinnosti nebo dokonce k předčasnému zničení.

5.2 PS REGULÁTOR [15] [10]

Jedná se o diskrétní ekvivalent spojitého PI regulátoru. Algoritmus v regulátoru pouze simuluje chování spojitého členu. Základní rovnice PI regulátoru je dána vztahem:

$$u(t) = K[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau] \qquad (4)$$

kde: K - zesílení regulátoru

T_I - integrační konstanta

e(t) - regulační odchylka

u(t) - akční veličina

„Proporciální zesílení odpovídá přirozené akci regulátoru, integrační konstanta byla zavedena z důvodů potlačení trvalé ustálené odchylky.” ([15], strana 14).

Mohlo by se zdát, že použití PID regulátoru neboli jeho diskrétního ekvivalentu PSD, by zlepšilo vlastnosti přechodového děje a regulace obecně. Protože použití derivační konstanty v regulátoru zrychluje přechodový děj a zlepšuje stabilitu. Ovšem použití derivační konstanty, při tak velikých časových konstantách soustavy jaké se objevují při regulaci domů a bytů, nemá žádný příznivý účinek na regulaci. Tato skutečnost byla ověřena v Matlabu při simulacích na modelu místnosti z práce [17].

Pro vytvoření číslicového algoritmu PI regulátoru je nutné jej převést do diskrétní podoby. Převod spočívá v nahrazení integrační složky za sumační složku. Výsledná rovnice udávající výpočet akčního zásahu PS regulátoru má tvar:

$$u(k) = K[e(k) + \frac{T_S}{T_I}\sum_{i=1}^{k} e(i)] \qquad (5)$$

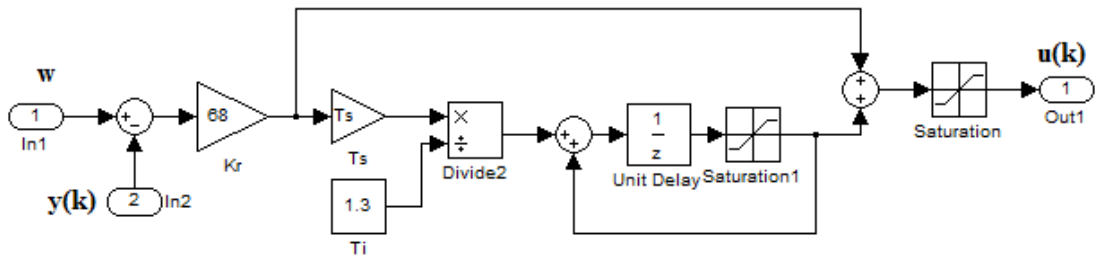
kde: T_S – perioda vzorkování

„Protože při numerické integraci počítáme plochu pod průběhem regulační odchylky jako součet obdélníků, násobíme sumaci periodou vzorkování. Samozřejmou součástí jakéhokoliv číslicového regulátoru musí být omezení akční veličiny (jeho neexistence by způsobila přetečení proměnné D/A převodníku a chybný akční zásah) a z toho vyplývající omezení sumační složky.“ ([18], strana 37).

Omezení sumační složky pomocí saturace je nejjednodušší možný způsob a pro naše použití je dostačující. V návrhu nesmíme zapomenout na celkové omezení akčního zásahu tedy přidáním další saturace na výstup regulátoru.

5.2.1 Návrh regulátoru

Podle rovnice (5) jsme sestavili model PS regulátoru v programu Matlab-Simulink. Na obrázku 22 je znázorněn PS regulátor s omezením sumační složky. Regulátor je navržen tak, aby reguloval model místnosti z práce [17]. Konstanty regulátoru jsme zjistili experimentálně simulací na modelu místnosti.



Obr. 22: Simulační schéma PS regulátoru

Přenos regulátoru v Z-transformaci:

$$F_r(z) = K[1 + \frac{T_S z^{-1}}{T_I(1 - z^{-1})}] \qquad (6)$$

kde: K = 68

T_I = 1.3 s

T_S = 0.1 s

Vzorkovací periodu jsme zvolili T_S = 0.1 s tak, aby byl splněn vzorkovací (S-K) teorém. Protože je vypočítaný akční zásah z PS regulátoru převáděn pomocí relé s hysterezí na ON/OFF, není rozsah akčního zásahu PS regulátoru vázaný na hardwarové řešení regulátoru. Akční zásah je shora omezen na hodnotu 100 a zdola na 0. Tyto hodnoty jsme zvolili, proto aby bylo možné jednoduše vyjádřit akční zásah v procentech. Použité relé s hysterezí spíná při 50% akčního zásahu a vypíná při poklesu akčního zásahu pod 30%. Další způsob převodu akčního zásahu na ON/OFF je PWM (pulzní šířková modulace). Tento způsob má pro použití v regulaci teploty domů značnou nevýhodu, protože nedokáže reagovat na změnu žádané veličiny nebo na poruchu dříve než uplyne zvolená perioda.

Pro realizaci regulátoru je nutné naprogramovat algoritmus do mikroprocesoru. Při programování vycházíme ze stavové proměnné na zpožďovači. Ta je zvolena jako proměnná *sum* a představuje hodnotu sumy vytvářené postupným sčítáním hodnot vytvářených integrační složkou regulátoru. Výpočet akčního zásahu je poté dán součtem signálů procházejících větvemi regulátoru (obr. 22). [18]

$$U = K \cdot e + sum \tag{7}$$

$$sum = sum + \frac{KT_S}{T_I} e \tag{8}$$

kde: U - hodnota akčního zásahu

sum – stavová proměnná sumátoru

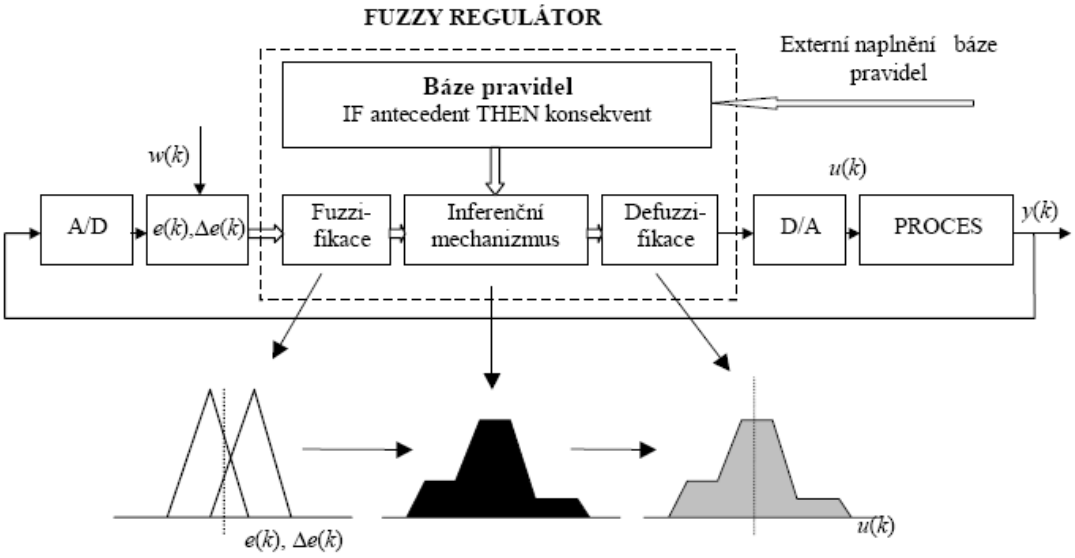
Zdrojový kód v jazyce Dynamic C pro výpočet akčního zásahu podle PS regulátoru:

```
float PS(float Reg_vel){
      e=W-Reg_vel;
      Akcni_Zasah= K*e+sum;
      sum = sum+K*(Ts/Ti)*e;
      if(Akcni_Zasah>10) Akcni_Zasah=10;
      if(Akcni_Zasah<0) Akcni_Zasah=0;
      if(sum>10) sum=10;
      if(sum<0) sum=0;
      return Akcni_Zasah;
      }
```

5.3 FUZZY REGULÁTOR [15] [13]

Pro pochopení činnosti Fuzzy regulátoru je nutné vysvětlit si několik pojmů a ukázat způsob zpracování odchylky, který je úplně jiný než u klasických regulátorů. Fuzzy regulátory používají tzv. Fuzzy logiku. *„Ta spočívá v rozšíření logických operátorů na fuzzy množiny. Teorie fuzzy množin spočívá v zavedení tzv. stupně příslušnosti prvku k množině, který může nabývat hodnot z intervalu <0, 1>. Fuzzy logika nám poskytuje jazyk s vlastní syntaxí a sémantikou, který nám umožňuje bezprostřední použití kvalitativně formulovaných znalostí a zkušeností o řešeném problému.“* ([15], strana 81).

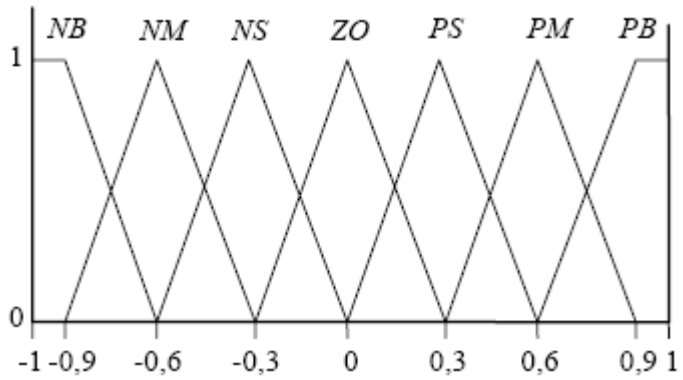
Použití fuzzy regulátoru spočívá ve využití fuzzy logiky a inferenčního mechanismu na vstupní proměnné. Základní struktura fuzzy regulátoru je znázorněna na obr. 23. Vzhledem k tomu, že regulátor je realizován číslicově je nutné použití A/D a D/A převodníků. Vstupem do regulátoru je regulační odchylka e(k) a její první diference Δe(k). *„Vstupní proměnné se v prvním kroku převedou do fuzzy množin. Fuzzifikovaná hodnota pak vstupuje do inferenčního mechanismu, který pracuje s bází pravidel. Výsledkem inferenčního mechanismu je fuzzy množina, ze které je v závěrečném kroku defuzzifikací určena velikost akčního zásahu. Nejčastější metoda fuzzy inference je metoda Min-Max.“* ([15], strana 85).



Obr. 23: Fuzzy regulátor ve zpětnovazebním zapojení [15]

5.3.1 Fuzzy PI regulátor s normalizovaným tvarem universa [15]

Obecně se dá říct, že nastavování fuzzy regulátorů je mnohem složitější než u klasických regulátorů. Použití fuzzy PI regulátoru s normalizovaným tvarem universa značně zjednoduší nastavování. Normalizace spočívá v tom, že rozsahy universa vstupních a výstupních proměnných jsou v intervalu <-1, 1> (obr. 24). Vstupní nebo výstupní proměnná je poté vynásobena konstantou, která vyjadřuje skutečný rozsah universa.



Obr. 24: Normalizované symetrické rozložení funkcí příslušností [15]

Vztah pro výpočet akčního zásahu fuzzy PI regulátoru v kroku k je:

$$u(k) = K\frac{MT}{T_I} D\left\{ F\left\{ \frac{T_I}{M}\Delta e(k) + \frac{1}{M} e(k) \right\} \right\} + u(k-1) \qquad (9)$$

kde: K - zesílení regulátoru

T_I - integrační konstanta

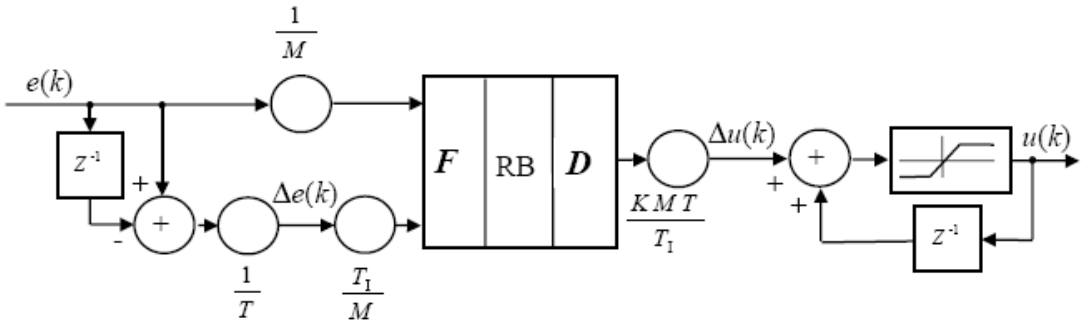
T - vzorkovací perioda

M – konstanta měřítka pro rozsah universa

D – operace defuzzifikace

F – operace fuzzifikace

„Fyzikální význam parametrů u fuzzy PI regulátoru zůstal zachován jako u PI regulátoru. Při nastavování parametrů u fuzzy PI regulátoru můžeme postupovat obdobně jako při nastavování parametrů u klasického PI regulátoru.“ ([15], strana 98).

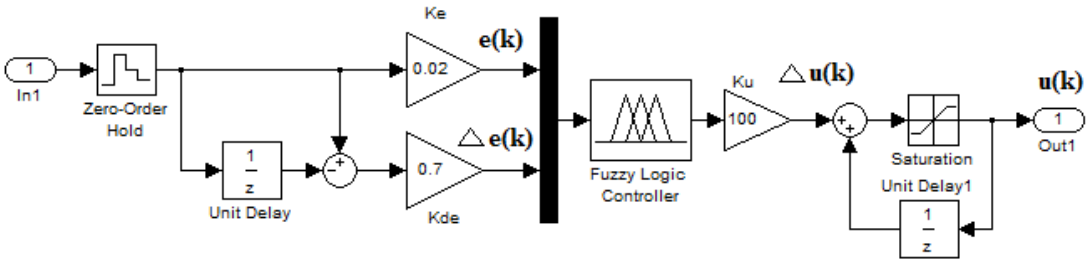


Obr. 25: Struktura fuzzy PI regulátoru s normalizovaným rozsahem universa [15]

5.3.2 Návrh regulátoru

Pro regulaci modelu místnosti [17] jsme zvolili fuzzy PI regulátor s normalizovaným tvarem universa, ze stejných důvodů jako byly uvedeny výše v kapitole 5.2. Fuzzy PI regulátor jsme navrhli pomocí fuzzy toolboxu v Matlabu. Regulátor jsme ponechali v základním nastavení, tedy typ regulátoru je mamdami. Metoda fuzzy inference je typu Min-Max a metoda defazzifikace je typu těžiště. Dále jsme definovali dvě vstupní proměnné a jednu výstupní proměnou. Pro všechny proměnné jsme použili normalizované symetrické rozložení funkcí příslušností (obr. 24). Výsledný model fuzzy PI regulátoru je znázorněn na obrázku 26. Hodnoty konstant jsme nastavili experimentálně simulací, tak aby regulátor reguloval model místnosti z práce [17].

Rozsah akčního zásahu jsme zvolili stejný jako u PS regulátor v kapitole 5.2. Tedy 0 – 100%. Použitý převod akčního zásahu na ON/OFF je také totožný jako u PS regulátoru v kapitole 5.2.



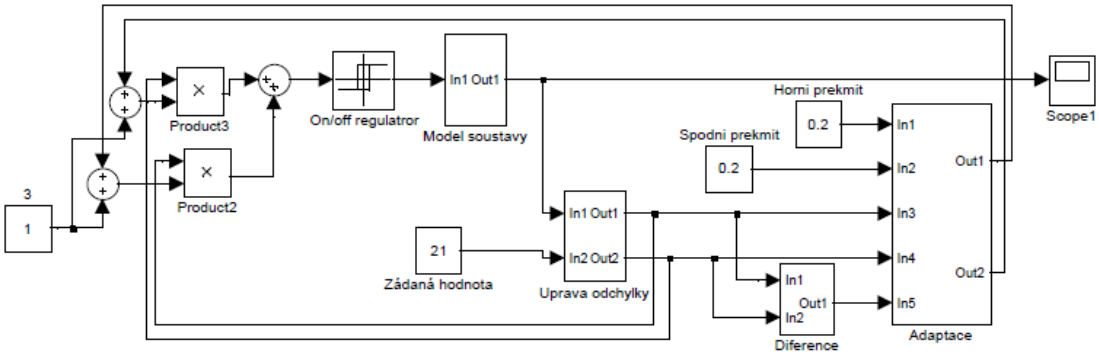
Obr. 26: Simulační schéma Fuzzy regulátoru

5.1 ON/OFF REGULÁTOR S AUTOMATICKÝM PŘENASTAVENÍM PARAMETRŮ

Při návrhu algoritmů řízení a regulace teploty pro model místnosti z práce [17], jsme uvažovali přesně nastavené hodnoty v modelu. Je tedy zřejmé, že pokud bychom parametry modelu změnili, navržené algoritmy by nedokázali kvalitně regulovat takovýto model. Požadujeme tedy, aby regulační algoritmus byl univerzální pro použití v jakékoli soustavě. Při regulaci teploty v domě pokojovým termostatem je tento požadavek velmi důležitý. Proto jsme navrhli algoritmus, který se dokáže soustavě přizpůsobit. Zvolili jsme ON/OFF regulátor pro jeho jednoduchost a výsledky srovnatelné s pokročilými algoritmy řízení PI a fuzzy. Princip tohoto algoritmu, tak jak jsme jej uvedli kapitole 5.1, spočívá ve volbě hystereze spínání a vypínání zdroje tepla. Vlivem rozdílné setrvačnosti soustavy teplota obvykle kolísá ve větším rozsahu teplot, než je právě zvolená hystereze. Tento nepříjemný jev jsme se rozhodli potlačit automatickou změnou právě horní a dolní hranice spínání kotle.

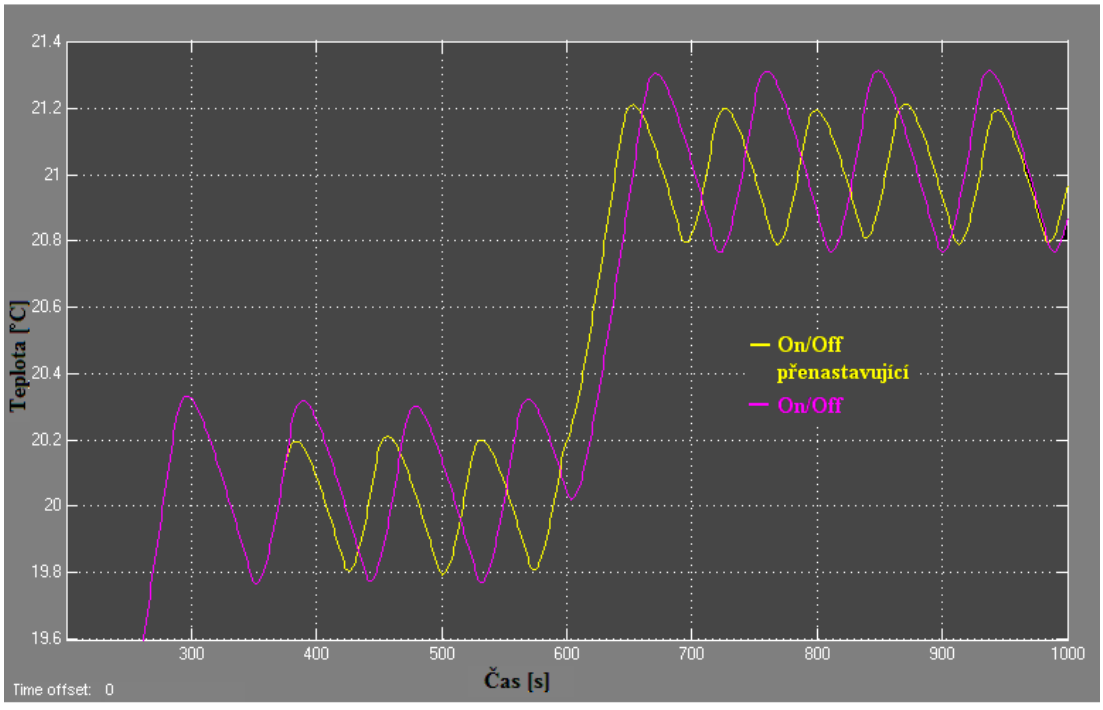
5.1.1 Princip regulátoru

Přenastavení mezí spínání kotle je realizováno algoritmem, jehož hlavní složkou je integrátor, který zajistí splnění přibližně nulové odchylky maximálního rozkmitu teploty od zvolených mezí. Uživatel tedy zvolí rozkmit hodnot teploty v místnosti. Zpočátku regulace jsou tyto meze přímo použity pro spínání kotle. Tedy např. při volbě hystereze 0,6 °C jsou meze spínání kotle ± 0,3 °C od požadované teploty. Pokud ovšem dojde během regulace vlivem setrvačnosti k jejich překročení jsou meze spínání sníženy tak, aby maximální/minimální hodnota teploty v místnosti co nejpřesněji odpovídala zvolené hysterezi.



Obr. 27: Simulační schéma ON/OFF regulátoru s přenastavením parametru

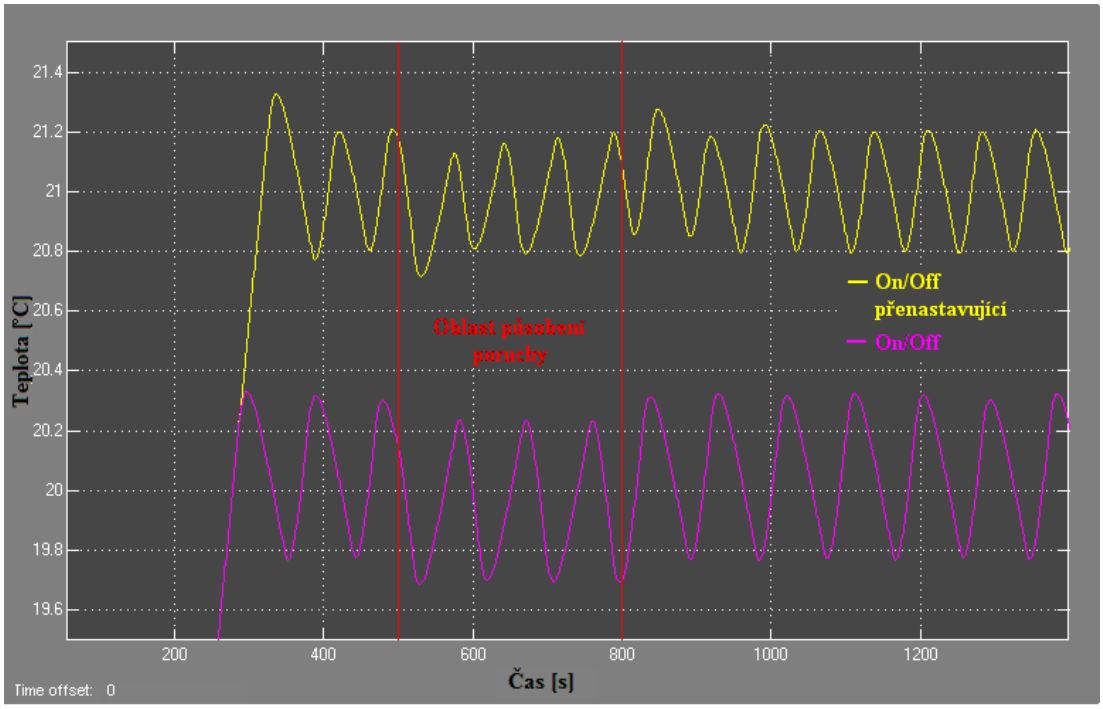
Činnost algoritmu zajišťujícího přenastavení bude vysvětlena na simulačním schématu na obrázku 27. V bloku úprava odchylky nejprve dojde k výpočtu odchylky a rozdělení jejího průběhu na kladnou a zápornou složku. Tyto průběhy poté vstupují do bloku diference, který na základě nulové diference (nahrazení derivace) určí okamžiky, kdy dosahuje odchylka lokálního maxima. V bloku adaptace je pak v okolí těchto maxim porovnávána hodnota odchylky od zvolených mezí a integrací této odchylky je vytvářen akční zásah, který provede požadované přestavení parametrů tak, aby byla odchylka 0. Přenastavení probíhá v tomto modelu formou zesílení/zeslabení kladné nebo záporné části odchylky.



Obr. 28: Regulovaná veličina při skokové změně řízení, venkovní teplota 4 °C

Na obrázku 28 vidíme průběh regulované veličiny při skokové změně řízení ON/OFF regulátoru a ON/OFF přenastavujícího se regulátoru. Pro ukázku funkčnosti navrženého algoritmu jsme změnili nastavenou setrvačnost v modelu místnosti z práce [17] z 1,6 s na 10 s. Hystereze obou srovnaných regulátorů jsou nastaveny na 0,4 °C. Vidíme tedy, že regulovaná veličina při použití ON/OFF regulátoru kmitá v rozsahu větším než je nastavená hystereze. Při použití ON/OFF přenastavujícího kmitá regulovaná veličina po jednom překmitu v nastavených mezích hystereze.

Na obrázku 29 vidíme průběh regulované veličiny při skokovém působení poruchy ON/OFF regulátoru a ON/OFF přenastavujícího se regulátoru. Z obrázku vidíme, že ON/OFF přenastavující se regulátor reaguje na poruchu vzniklou v obvodu a kompenzuje ji.



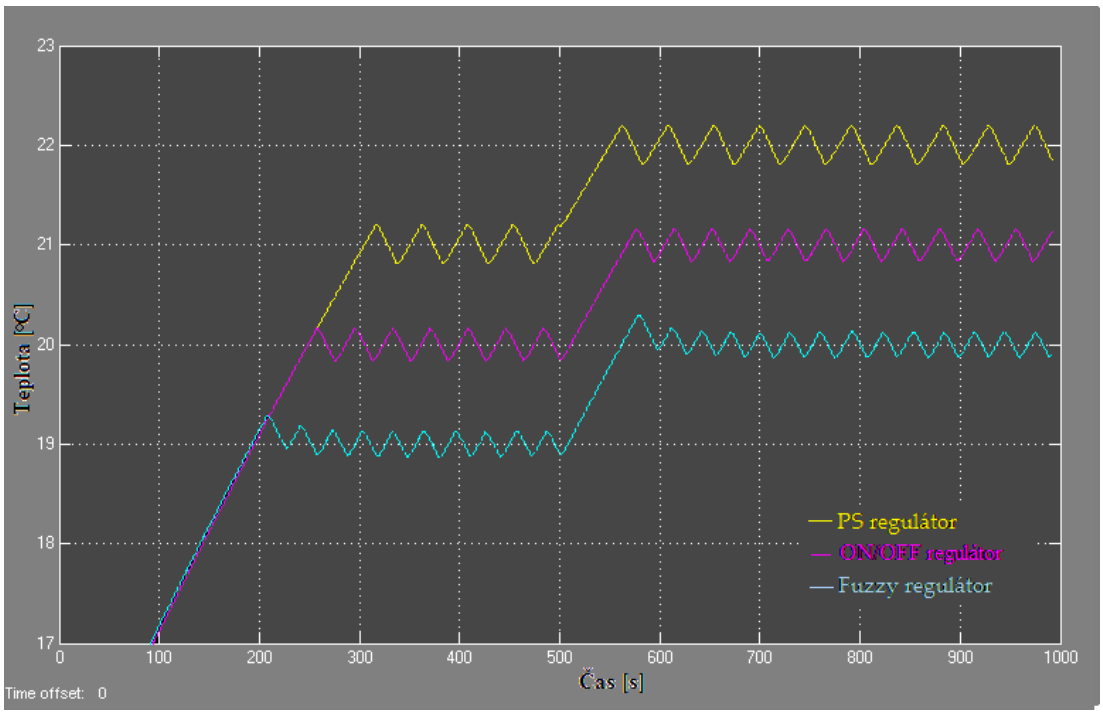
Obr. 29: Průběh regulované veličiny při skokovém působení poruchy, venkovní teplota 4 °C

6. SROVNÁNÍ ALGORITMŮ ŘÍZENÍ A REGULACE

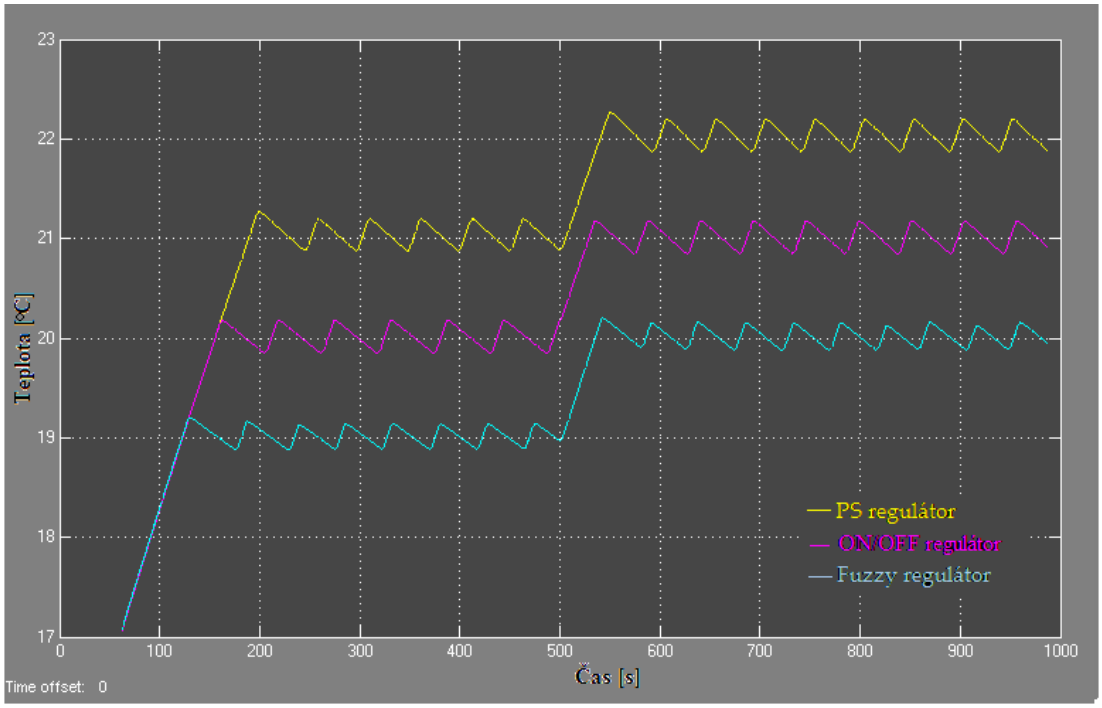
Pro výběr nejlepšího algoritmu pro řízení a regulaci teploty v domě je nutné provést srovnání všech navrhnutých algoritmů. Mezi základní požadavky na regulaci patří věrné sledování řízení a kompenzace poruch. Proto jsme při srovnání algoritmů vycházeli z těchto dvou požadavků. Mezi další kritéria, pro výběr nejlepšího algoritmu, můžeme zařadit ekonomický faktor regulace a realizovatelnost regulátoru.

Protože termodynamický model místnosti [17] dovoluje změnu venkovní teploty, provedli jsme srovnání regulátorů při různých venkovních teplotách, aby bylo zřejmé, jak se projeví vliv venkovní teploty.

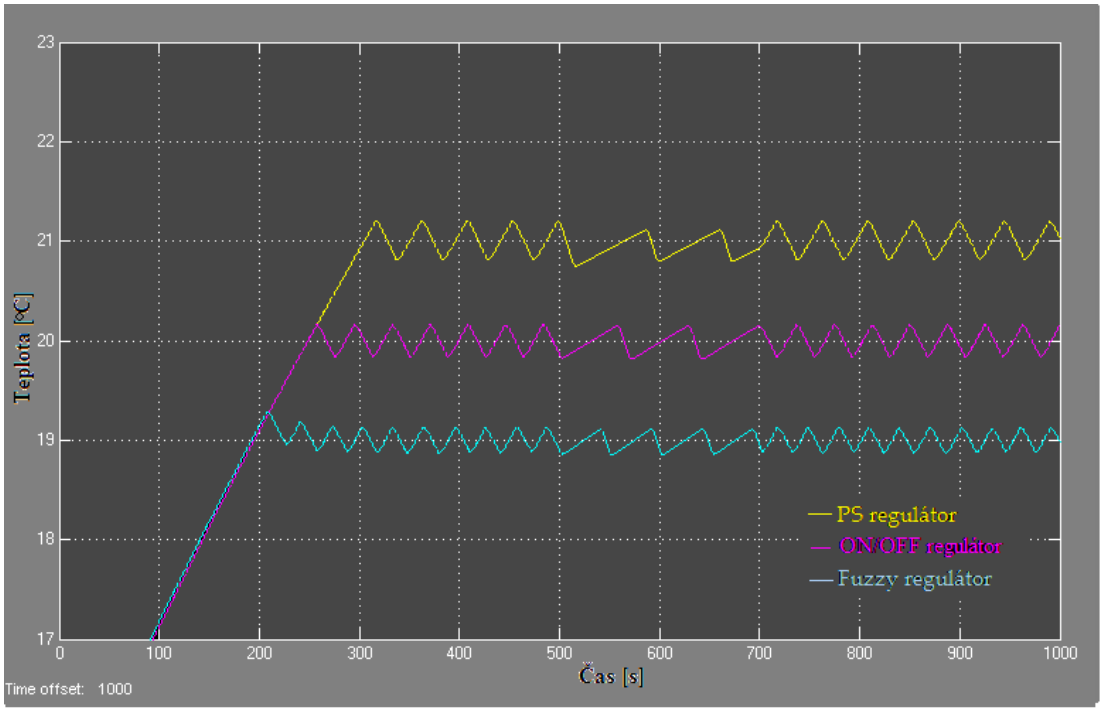
Na osách grafů je zobrazena závislost výstupní veličiny (teplota vzduchu v místnosti) na čase. Zobrazený čas v simulaci je uveden v sekundách, ovšem jedna sekunda v simulaci odpovídá 1 minutě v reálném objektu. Pro lepší srovnání algoritmů jsme je zobrazili pod sebou, takže každý reguluje na jinou požadovanou teplotu. Regulátory jsou nastaveny tak, aby jejich regulovaná veličina kmitala kolem požadované veličiny s hysterezí 0.4 °C.



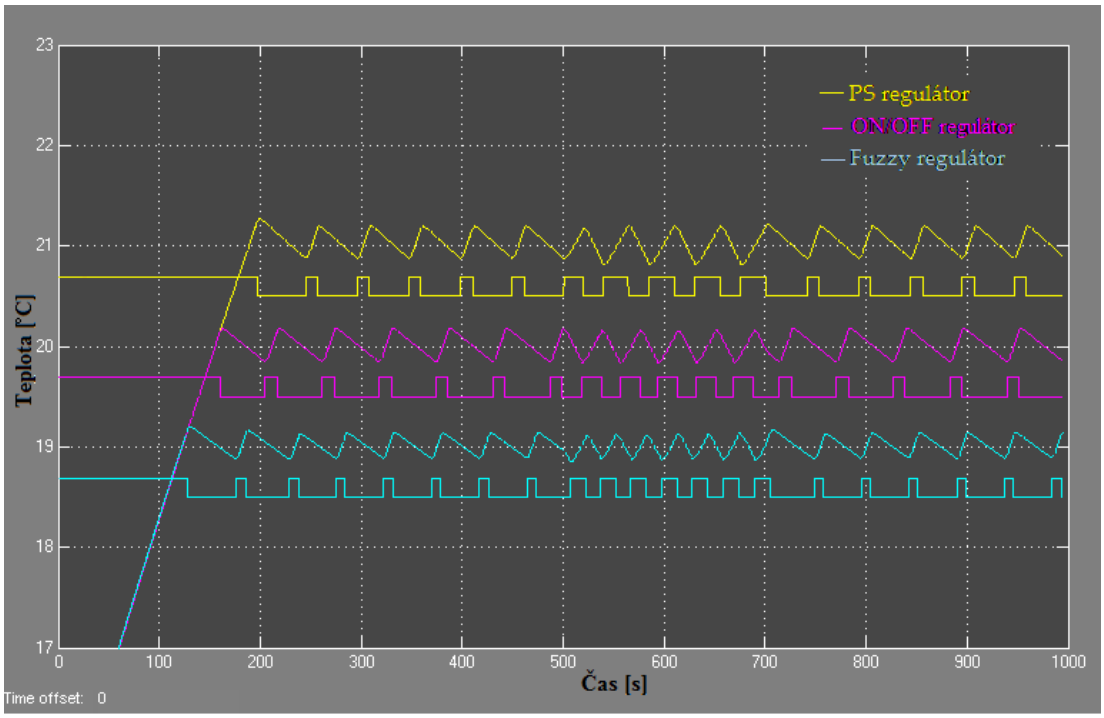
Obr. 30: Regulovaná veličina při skokové změně řízení, venkovní teplota -5 °C



Obr. 31: Regulovaná veličina při skokové změně řízení, venkovní teplota 10 °C



Obr. 32: Průběh regulované veličiny při skokovém působení poruchy, venkovní teplota -5 °C



Obr. 33: Průběh regulované veličiny při skokovém působení poruchy, venkovní teplota 10 °C

Na obrázcích 32 a 33 jsou znázorněny průběhy regulovaných veličin regulátorů při skokovém působení poruchy, pro venkovní teploty -5 °C a 10 °C. Porucha začíná působit v čase 500s a skončí v čase 700s. Na obrázku 33 je znázorněn akční zásah regulátorů. Je upraven tak, aby byl dobře zobrazitelný na grafu.

Porovnáním průběhů na obrázcích 30 až 33 vidíme, že velikost venkovní teploty má zásadní vliv na rychlost regulace. Můžeme tedy říct, že čím větší venkovní teplota tím rychlejší regulace. Dále je zřejmé, že průběhy regulovaných veličin u všech použitých regulátorů jsou si velmi podobné. Vliv působení poruchy všechny regulátory kompenzují a na změnu požadované veličiny reagují okamžitě. Z těchto výsledků vyplývá, že použití pokročilých algoritmů řízení (PS regulátor a Fuzzy PI regulátor) nepřineslo do regulace teploty v místnosti podle modelu [17] žádné zlepšení. Je tedy zřejmé, že použití jednoduché ON/OFF regulace je pro naši soustavu zcela dostačující. U fuzzy regulátoru není toto konstatování přesné, protože použitím sofistikovanější fuzzy regulace bychom mohli dosáhnout kvalitnější regulace než u ON/OFF regulace.

7. REALIZACE PŘÍPRAVKU PRO IMPLEMENTACI ALGORITMU

7.1 POPIS HARDWARU

K ověření funkčnosti jednoho z navržených algoritmů jsme měli vytvořit termostat v tzv. vývojové verzi. K dispozici jsme měli vývojovou desku RABBIT 2000^{TM} TCP/IP DEVELOPMENT KIT od firmy Rabbit Semiconductor [7].

K této desce jsme připojili displej MC2004B-SYL/H pro zobrazování informací aktuální teploty, žádané teploty, akčního zásahu PS regulátoru a informace o stavu zdroje tepla (Obr. 34). Přenos dat na displej je 4 bitový. Podrobné informace k displeji najdete v literatuře [3], [4].

Měření teploty zajišťuje připojené inteligentní čidlo teploty DS18B20, které komunikuje s mikroprocesorem po 1-Wire sběrnici. Podrobné informace o tomto čidlu najdete v literatuře [2].

Ovládací tlačítka pro nastavení žádané teploty jsou připojena přímo k mikroprocesoru, využili jsme demonstrační desku dodanou k vývojovému kitu Rabbit2000^{TM}. Indikační LED dioda, která signalizuje zapnutí/vypnutí topení (žárovky) byla také připojena přímo k mikroprocesoru na pin 7 portu B.

Obr. 34: Zobrazení informací z procesu na displeji

7.1.1 Vlastnosti mikroprocesoru RABBIT 2000™ [7]

- 256kB program FLASH, 128kB SRAM
- 7x8bit časovačů/čítačů,
- 20x10bit ADC
- 3xUSART, RS-232, RS-485
- Oscilátor 18.432MHz
- Watch-dog, úsporné režimy

7.2 POPIS SOFTWARU

Zdrojový kód pro mikroprocesor Rabbit 2000™ jsme psali v programovacím prostředí Dynamic C, což je vývojové prostředí a také programovací jazyk vyvinutý přímo firmou Rabbit Semiconductor pro programování jejich mikroprocesorů. Tento jazyk je velmi podobný klasickému programovacímu jazyku C, ovšem vyskytují se v něm různé výjimky a změny. Např. při vkládání knihoven se nepoužívá #include, ale #use, dalším rozdílem je, že Dynamic C nedovoluje inicializaci proměnných již při deklaraci proměnných.

Při seznamování s tímto prostředím jsme využili některé vzorové příklady volně přidávané ke každému prostředí Dynamic C. Dále jsme vycházeli z datasheetů k použitým součástkám a z manuálů k vývojovému kitu. Manuály k programovému prostředí a vývojovému kitu jsou uvedeny v literatuře [5], [8].

7.2.1 Popis některých důležitých funkcí v programu

zmer ()

Funkce zajišťuje změření a převod teploty na binární číslo. Ve funkci se postupně volají všechny funkce potřebné k inicializaci sběrnice 1-Wire a převodu informace o teplotě na číslo. Nejprve je volána funkce **ow_detect_presence(),** která inicializuje sběrnici a identifikuje čidlo. Protože je na sběrnici připojeno pouze jedno čidlo a není tedy nutné číst jeho adresu, je použita funkce **owSkipRom().** Dále voláme funkce **owConvertT()**, která vyšle příkaz čidlu na změření teploty. Po 750

ms je možné pomocí funkce **owReadScratchPad()** vyžádat poslání naměřené hodnoty. Ta je ve formě dvou bytů, které jsou přečteny pomocí **funkce ow_read_byte().** Poté jsou hodnoty rozděleny na celočíselnou a desetinou část. Podrobnosti o komunikace s čidlem teploty najdete v literatuře [2]. Celý kód funkce je uveden v příloze 1.

PS (float Reg_vel)

Funkce zajišťuje výpočet akčního zásahu podle algoritmu PS regulátoru, tak jak byl navrhnut výše v kapitole 5.2. Vstupní proměnná funkce je Reg_vel, ve které je uložena informace o naměřené teplotě. Funkce vrací programu hodnotu akčního zásahu v proměnné Akcni_Zasah. Celý kód funkce je uveden v příloze 1.

LCD_init ()

Funkce zajišťuje inicializaci displeje. Je nutné dodržet algoritmus uvedený výrobcem v literatuře [3]. Pomocí inicializace definujeme počáteční nastavení displeje. Např. typ přenosu dat (4 bit, 8bit), počet řádků, rozlišení segmentů, zapnutí/vypnutí kursoru. Ve funkci posíláme na displej příkazy, je tedy nutné nastavit signál RS (**iDataFlag = COMMAND;)**. Celý kód funkce je uveden v příloze 1.

Display (char *szp)

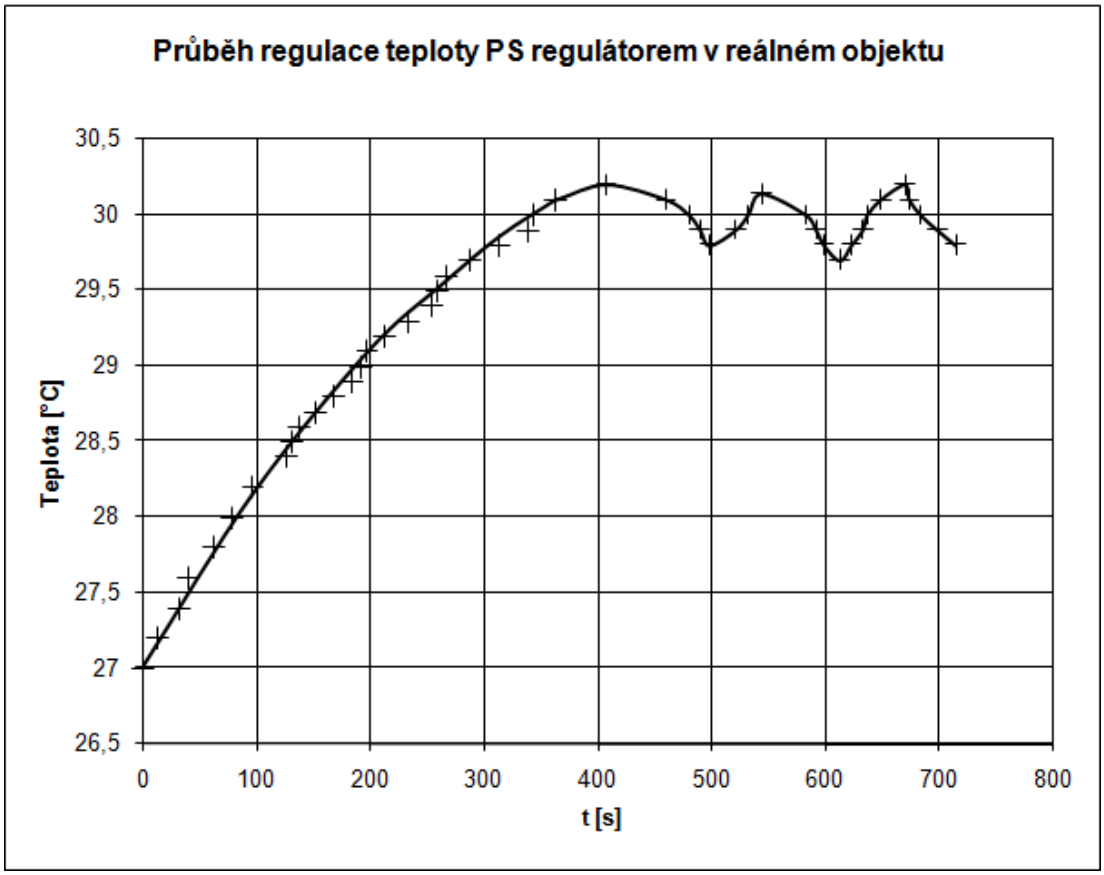
Funkce zajišťuje zobrazení informací uložených ve vstupní proměnné typu char na displej. Funkce postupně posílá na displej jednotlivé znaky. Protože používáme 4 bitový přenos dat, je nutné posílat data na displej nadvakrát. Voláním funkce **WriteByte (char cNib)** si rozdělíme jeden znak na půlky a funkcí **WriteNibble (char cNib)** se postupně pošlou obě poloviny znaku na displej. Celý kód funkce je uveden v příloze 1.

Plovoucí průměr

Při realizaci jsme se setkali s problémem kolísání změřené teploty v řádech desetin stupně celsia, tento nepříznivý jev je způsoben chybou měření čidla, zaokrouhlováním při počítání s teplotou, ale hlavně prouděním vzduchu kolem čidla. Tuto chybu lze odstranit naprogramování plovoucího průměru. Celý kód je uveden v příloze 1.

8. VYHODNOCENÍ VÝSLEDKŮ MĚŘENÍ V REÁLNÉM OBJEKTU

Pomocí papírové krabice o rozměrech 110x200x310mm jsme simulovali reálný objekt. V této krabici byla vložena 10W žárovka, která simulovala topení v objektu. Sestavený přípravek s implementovaným PS algoritmem plnil funkci regulátoru. Je zřejmé, že tento reálný objekt nemá stejné časové konstanty jako termodynamický model místnosti z práce [17] podle, kterého byl PS regulátor nastaven. Bylo tedy nutné experimentálně přenastavit konstanty regulátoru tak, aby správně reguloval tuto soustavu. Musíme však podotknout, že PS regulátor s konstantami nastavenými na termodynamický model místnosti dokázal regulovat teplotu v objektu podobně jako na modelu, avšak s větším rozkmitem regulované veličiny asi 0,8 °C.



Obr. 35: Průběh regulované veličiny v reálném objektu

Na obrázku 35 vidíme průběh regulované veličiny v reálném objektu, požadovaná veličina byla nastavena na 30 °C. Rozkmit teploty je 0,4 °C. Z výsledků vyplývá, že implementovaný PS algoritmus je funkční.

9. ZÁVĚR

V úvodní části práce je kompletní návrh pokojového termostatu použitelného k regulaci teploty v domech nebo bytech. Návrh zohledňuje požadavky kladené na pokojové termostaty a také odstraňuje, některé nedostatky vyráběných termostatů. Řidící částí navrženého termostatu je mikroprocesor AVR ATmega64-16AU, který svými vlastnostmi vyhovuje požadavků kladeným na termostaty, ať už je to rychlost zpracování nebo velikost paměti. Měření teploty zajišťuje inteligentní čidlo DS1631, které zaručí dostatečně přesné měření teploty. Zobrazování informací z regulovaného procesu je řešeno pomocí alfanumerického maticového displeje MC2004B-SYL/H. Výstupní členem navrženého termostatu je relé, které spíná zdroj tepla. Výsledkem návrhu je elektrické schéma, podle kterého by bylo možné termostat vyrobit.

Dále se v práci zabýváme návrhem algoritmů pro regulaci teploty v místnostech. V práci [17] byl navržen termodynamický model místnosti v programu MATLAB. Na základě tohoto modelu jsme simulovali regulovanou soustavu, na kterou jsme navrhli tři regulátory. Jedním z těchto regulátorů je klasický ON/OFF regulátor. Dále jsou to dva pokročilé algoritmy řízení, jedná se o PS regulátor a Fuzzy regulátor. Srovnáním těchto tří algoritmů na termodynamickém modelu jsme dospěli k závěru, že použití pokročilých algoritmů řízení nepřineslo do regulace teploty v místnosti žádné zlepšení. Je tedy zřejmé, že použití jednoduché ON/OFF regulace je pro naši soustavu zcela dostačující.

Obecně můžeme říct, že od regulačního algoritmu požadujeme, aby byl univerzální, tedy použitelný v jakékoli soustavě. Při regulaci teploty v domě pokojovým termostatem je tento požadavek obzvláště důležitý. Proto jsme navrhli ON/OFF regulátor, který se dokáže soustavě přizpůsobit. Tento regulátor vykazuje oproti klasickému ON/OFF regulátoru, mnohem lepší výsledky tím, že regulovaná veličina kmitá v přesně nastaveném pásmu hystereze.

V závěru práce byl pro ověření funkčnosti PS algoritmu vytvořen termostat v tzv. vývojové verzi. Do vývojové desky s mikroprocesorem Rabbit 2000™ jsme implementovali PS algoritmus. K této desce bylo připojeno čidlo teploty a LCD displej. Měřením v reálném objektu jsme ověřili funkčnost termostatu.

10. SEZNAM ZKRATEK

Zkratka	Význam zkratky	Zkratka	Význam zkratky
LED	Light Emitted Diode	RISC	Reduced Instruction Set Computer
PID	Proporcionálně-Integračně Derivační	SPI	Serial Peripheral Interface
NTC	Negative Temperature Coefficient	PWM	Pulse Width Modulation
PTC	Positive Temperature Coefficient	UART	Universal Asynchronous Receiver and Transmitter
A/D	Analog/Digital	DI	Digital Input
ROM	Read-Only Memory	DO	Digital Output
EPROM	Erasable Programmable Read-Only Memory	MOSI	Master Output Slave Input
EEPROM	Electronically Erasable Programmable Read-Only Memory	MISO	Master Input Slave Output
SRAM	Static Random Access Memory	LCD	Liquid Crystal Display
RAM	random-access memory		

11. LITERATURA

[1] ANONYM.: *Katalogový list DS1631*[online], Dallas Semiconductor, Dostupné z: < http://datasheets.maxim-ic.com/en/ds/DS1631-DS1731.pdf>

[2] ANONYM.: *Katalogový list DS18B20*[online], Dallas Semiconductor, Dostupné z: < http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf>

[3] ANONYM.: *Katalogový list HD44780* [online], HITACHI SEMICONDUCTOR, Dostupné z: < http://pdf1.alldatasheet.com/datasheet-pdf/view/63673/HITACHI/HD44780.html >

[4] ANONYM.: *Katalogový list MC2004B-SYL/H*[online], EVERBOUQUET INTERNATIONAL, Dostupné z: < http://www.gme.cz/_dokumentace/dokumenty/513/513-125/dsh.513-125.1.pdf>

[5] ANONYM.: *Dynamic C User's manual* [online], Rabbit Semiconductor, Dostupné z: < http://www.rabbit.com/products/rab2000/>

[6] ANONYM.: *Katalogový list ATmega64-16AU* [online], ATMEL, Dostupné z: <http://www.atmel.com/dyn/resources/prod_documents/doc2490.pdf>

[7] ANONYM.: *Katalogový list Rabbit 2000* [online], Rabbit Semiconductor, Dostupné z: < http://www.rabbit.com/products/rab2000/>

[8] ANONYM.: *Rabbit 2000™ TCP/IP DEVELOPTMENT KIT-GETTING STARTED* [online], Rabbit Semiconductor, Dostupné z: <http://www.rabbit.com/products/rab2000/>

[9] BEJČEK L.: *Měření neelektrických veličin*. Skriptum, VUT Brno, 1988

[10] BLAHA P., VAVŘÍN P.: *Řízení a regulace 1*, VUT Brno: 2005, s. 54-214

[11] HRBÁČEK J.: *Komunikace mikrokontroléru s okolím 1. Díl*, Praha, BEN-technická literatura:1999, s. 159, ISBN: 80-86056-42-2

[12] HRBÁČEK J.: *Komunikace mikrokontroléru s okolím 2. Díl*, Praha, BEN-technická literatura:200, s. 151, ISBN: 80-86056-73-2

[13] JURA P.: *Základy Fuzzy logiky pro řízení a modelování*, VUT Brno:2003, s. 1-132, ISBN: 80-214-2261-0

[14] NOVOTNÝ Z.: *Relé s ekonomickým provozem*[online], HW server, 200-01-01, Dostupné z. http://hw.cz/Teorie-a-praxe/Navrhy-vyvojare/ART1029-Rele-s-ekonomickym-provozem.html

[15] PIVOŇKA P.: *Číslicová řídící technika, VUT* Brno: 2003, s. 151

[16] RIPKA P., ĎADO S., KREIDL M., NOVÁK J.: *Senzory a převodníky*, Praha, Vydavatelství ČVUT 2005, s. 136, ISBN: 80-01-03123-3

[17] VALÍČEK J., PADĚRA Z.: *Pokojový termostat nové generace*, Semestrální práce, UAMT 2008

[18] VELEBA V.: *Číslicová řídící technika (počítačové cvičení).* VUT Brno: 2005, s. 77

SEZNAM PŘÍLOH

Příloha 1

```
#class auto
char data_lo, data_hi, p_teplota[5],p_desetiny[5],p_Akcni[5], pW_des[6], p_W[6];
int desetiny, teplota, ii, n, Akcni, tep, des, W_des, pom_W, pom_X, W_tep;
unsigned int up, down;
float Akcni_Zasah, Reg_vel, sum, e, Reg_pred, prumer[6], soucet, X, W;
const float K=7.2, Ti=3, Ts=0.1;
const int Tt=1;

//funkce pro časové zpoždění v mikrosekundách
void UsDelay ( int iDelay ){
      auto int i;
      iDelay /= 11;
      for ( i=0; i<iDelay; i++ );
}

//funkce pro časové zpoždění v milisekundách
void msDelay(unsigned int delay){
      auto unsigned long done_time;
      done_time = MS_TIMER + delay;
      while( (long) (MS_TIMER - done_time) < 0 );
}

//funkce pro nastavení pinu 7 na portu D na nulu
void TX (){
      WrPortI(PDDDR, &PDDDRShadow, 0xFF); // nastavý bity na portu D jako výstupní
      WrPortI(PDDCR, &PDDCRShadow, 0x00); // nastavý všechny bity na portu D na
                                          //open drain
      WrPortI(PDFR, &PDFRShadow, 0x00);  //  nastavý všechny bity portu D na
                                          //normální funkci
      BitWrPortI(PDDR, &PDDRShadow, 0, 7); // pošle na pin 7 portu D nulu
}

//funkce pro nastavení portu D jako vstupní
void RX (){
      WrPortI(PDDDR, &PDDDRShadow, 0x00);
}

//funkce čtení, čte pin, na kterém je připojeno čidlo teploty
unsigned int READ(){
      unsigned int x;
      WrPortI(PDDDR, &PDDDRShadow, 0x00);      // nastavý port D jako vstupní
      x=BitRdPortI(PDDR,7);                    // čtení z pinu 7 na portu D
      return x;
}

//funkce provede reset a test prezence ds-18b20 na sbernici
unsigned int ow_detect_presence(void) {
      unsigned int out;
      RX();                                 // vychozi stav sbernice
      UsDelay(1000);                        // časové zpoždění pro ustaleni
      TX();                                 // stáhnutí sběrnice k nule
      UsDelay(480);                         // cas pro prikaz reset
      RX();                                 // uvolneni sbernice
      UsDelay(70);                          // cekani na potvrzeni čidlem
      if(READ()) out=0;
      else out=1;                           // pokud je detekovana log.1, tak čidlo
                                            // na sbernici neni
      UsDelay(420);                             // pauza pred dalsi komunikaci
```

```
	return out;                          // vrati stav 1=čidlo nalezeno, 0=čidlo
                                         // nenalezeno
}



//funkce pro poslání log.1 na sbernici
void ow_write_one(void) {
	WrPortI(PDDDR, &PDDDRShadow, 0xFF); // nastavý bity na portu D jako výstupní
	WrPortI(PDDCR, &PDDCRShadow, 0x00); // nastavý všechny bity na portu D na
                                         //open drain
	WrPortI(PDFR, &PDFRShadow, 0x00);   // nastavý všechny bity portu D na
                                         //normální funkci
	BitWrPortI(PDDR, &PDDRShadow, 0, 7); // pošle na pin 7 portu D nulu
	WrPortI(PDDDR, &PDDDRShadow, 0x00); // uvolneni sbernice
	UsDelay(64);                         // pauza pred dalsi komunikaci
}

//funkce pro poslání log.0 na sbernici
void ow_write_zero(void) {
	TX();                          // stáhnutí sběrnice k nule
	UsDelay(60);                   // pauza definujici log.0
	RX();                          // uvolneni sbernice
}

//funkce pro přečtení jednoho bit ze sbernice
unsigned int ow_read_bit(void) {
	unsigned int out;
	WrPortI(PDDDR, &PDDDRShadow, 0xFF); // nastavý bity na portu D jako výstupní
	WrPortI(PDDCR, &PDDCRShadow, 0x00); // nastavý všechny bity na portu D na
                                         //open drain
	WrPortI(PDFR, &PDFRShadow, 0x00);   // nastavý všechny bity portu D na
                                         //normální funkci
	BitWrPortI(PDDR, &PDDRShadow, 0, 7);     // pošle na pin 7 portu D nulu
	WrPortI(PDDDR, &PDDDRShadow, 0x00);      // uvolneni sbernice
	if(READ()) out=1;
	else out=0;                          // test stavu sbernice, vlastni cteni
	UsDelay(55);                         // pauza pred dalsi komunikaci
	return out;                          // prectena hodnota, 1 nebo 0
}

//funkce prijme ze sbernice jeden byte. Prijima jako prvni LSB.
char ow_read_byte(void) {
	unsigned int out,n;
	n=8;
	out=0;
	while(n--) {
		out >>= 1;                           // bitovy posuv doprava
		if(ow_read_bit()) out |= 0x80;       // nastaveni nejvyssiho bitu na 1
	}
	return out;                              // vrací přečtený byte
}

//funkce pro poslání příkazu CONVERT T (0x44)
void owConvertT(void)
{
	ow_write_zero();        // volání funkce ow_write_zero
	ow_write_zero();
	ow_write_one();         // volání funkce ow_write_one
	ow_write_zero();
	ow_write_zero();
```

```
	ow_write_zero();
	ow_write_one();
	ow_write_zero();
				//čekání na konec převodu
	while (1){
	if(READ()) break;	// jakmile skončí ukončí se funkce
	}
}

//funkce pro poslání příkazu SKIP ROM (0xCC)
void owSkipRom(void)		// na sběrnici je připojeno jen jedno čidlo
{
	ow_write_zero();	// volání funkce ow_write_zero
	ow_write_zero();
	ow_write_one();		// volání funkce ow_write_one
	ow_write_one();
	ow_write_zero();
	ow_write_zero();
	ow_write_one();
	ow_write_one();
}

//funkce pro poslání příkazu READ SCRATCHPAD (0xBE)
void owReadScratchPad(void)
{
	ow_write_zero();	// volání funkce ow_write_zero
	ow_write_one();		// volání funkce ow_write_one
	ow_write_one();
	ow_write_one();
	ow_write_one();
	ow_write_one();
	ow_write_zero();
	ow_write_one();
}

// funkce, která změří teplotu
void zmer(void) {
	ow_detect_presence();		// volání funkce ow_detect_presence
	owSkipRom();			// volání funkce owSkipRom
	owConvertT();			// volání funkce owConvertT
	msDelay(750);			// časové zpoždění pro změření teploty
	ow_detect_presence();		// volání funkce ow_detect_presence
	owSkipRom();			// volání funkce owSkipRom
	owReadScratchPad();		// volání funkce owReadScratchPad
	data_lo=ow_read_byte();		// čtení 1.bytu ze scratchpadu čidla = spodni
					//byte teploty
	data_hi=ow_read_byte();		// čtení 2.bytu ze scratchpadu čidla = horni
					//byte teploty
	teplota=((data_hi & 0x0F) << 4)|((data_lo & 0xF0)>> 4);//uložení teploty v
							//celých číslech
	desetiny=(data_lo & 0x0F)*0.625+0.5; //uložení teploty za desetinou čárkou
}

//funkce pro výpočet akčního zásahu podle algoritmu PS regulátoru
float PS(float Reg_vel){
	e=W-Reg_vel;			// výpočet odchylky
	Akcni_Zasah= K*e+sum;		// algoritmus pro výpočet akčního zásahu
	sum = sum+K*(Ts/Ti)*e;		// přepočet stavové veličiny sumátoru

	if(Akcni_Zasah>10) Akcni_Zasah=10;  // omezení akční veličiny shora
```

```
      if(Akcni_Zasah<0) Akcni_Zasah=0;    // omezení akční veličiny zdola
      if(sum>10) sum=10;                  // omezení sumační složky shora
      if(sum<0) sum=0;                    // omezení sumační složky zdola
      return Akcni_Zasah;                 // funkce vrací hodnotu o akčním zásahu
}

#define DATA 0x04                   //makro pro zasílání dat
#define COMMAND 0x00                //makro pro zasílání instrukcí
int iDataFlag;



// funkce, která pošle první půl byte dat
void WriteNibble ( char cNib )
{
      cNib &= 0xF0;                       // vymaže spodní 4 bity
      cNib |= iDataFlag;                  // vloží datový příznak pro signál RS
      WrPortI ( PADR,&PADRShadow, cNib|0x03 ); // povolení E
      WrPortI ( PADR,&PADRShadow, cNib&0xFD ); // vymazání E
}

// funkce pošle bytu na displej
void WriteByte ( char cNib )
{
      auto int i;
      i = cNib;                     // naplnění pomocné proměnné
      cNib <<= 4;                   // posunutí vrchního půl bytu na spodní 4 bity
      WriteNibble ( i );            // pošle vrchní půl byte
      UsDelay(100);                 // pauza před dalsi komunikaci
      WriteNibble ( cNib );         // pošle spodní půl byte
      UsDelay (100);                // pauza před další komunikaci
}

//funkce pro inicializaci displeje
void LCD_Init ()
{
      WrPortI ( PADR,&PADRShadow, 0 );    // zapsání nuly na port A
      WrPortI ( SPCR,&PADRShadow, 0x84 ); // nastavení portu A jako výstupní
      iDataFlag = COMMAND;          // nastavení signálu RS pro zasílání instrukcí
      msDelay ( 1500 );             // zpoždění nutné pro resetování displeje
      WriteByte ( '\B00100000' );         // zvolení 4 bitového režimu
      UsDelay (100);
      WriteByte ( '\B11000010' );   // zvolení 4 bitového režimu, 2 řádků, 5x8
      WriteByte ( '\B11100000' );   // zapnutí displeje a kursoru
      WriteByte ( '\B00010000' );   // vynulování displeje
      WriteByte ( '\B00110000' );   // posun zobrazení a kurzoru
}

//funkce pošle data na displej
void Display ( char *szp )
{
      iDataFlag = DATA;             // nastavení signálu RS pro zasílání dat
      while ( *szp ) WriteByte ( *szp++ ); // cyklus, který postupně pošle na
                                           //displej všechna data v proměnné szp
}

//funkce pro vymazání displeje
void Clear ( void )
{
      iDataFlag = COMMAND;          // nastavení signálu RS pro zasílání instrukcí
```

```
      WriteByte ( 0x01 );           // vymaže displej
      msDelay (3);
}

//funkce pro nastavení kurzoru na druhý řádek
void Line2 ( void )
{
      iDataFlag = COMMAND;          // nastavení signálu RS pro zasílání instrukcí
      WriteByte ( 0xC0 );           // nastavení RAM adresy na řádek 2
      msDelay (3);
}

//funkce pro nastavení kurzoru na třetí řádek
void Line3 ( void )
{
      iDataFlag = COMMAND;          // nastavení signálu RS pro zasílání instrukcí
      WriteByte ( 0x94 );           // nastavení RAM adresy na řádek 3
      msDelay (3);
}

//funkce pro nastavení kurzoru na čtvrtý řádek
void Line4 ( void )
{
      iDataFlag = COMMAND;          // nastavení signálu RS pro zasílání instrukcí
      WriteByte ( 0xD4 );           // nastavení RAM adresy na řádek 4
      msDelay (3);
}

// hlavní funkce
void main(void) {
      LCD_Init ();                        // volání funkce LCD_Init
      Akcni_Zasah=0;                      // inicializace proměnných
      Reg_vel=0;
      sum=0;
      e=0;
      Reg_pred=28;
      W=28;
      ii=0;
      tep=0;
      des=0;
      pom_X=0;
  while(1)
  {
      zmer();                             // volání funkce zmer
      Reg_vel=teplota+(float)desetiny/10; // vyjádření hodnoty o teplotě v čísle
                                          // s plovoucí řádovou čárkou

      if(Reg_vel>Reg_pred+20){            // filtrace chyb teploty
            Reg_vel=Reg_pred;       // vzniklých při špatné komunikaci s čidlem
      }
      Reg_pred=Reg_vel;
                                          // realizace plovoucího průměru
      if(ii=0){
            for(n=1; n<6; n++){
                  prumer[n]=Reg_vel;
                  ii=1;
            }
      }
      n=5;
      prumer[n]=prumer[n-1];
```

```
        prumer[n-1]=prumer[n-2];
        prumer[n-2]=prumer[n-3];
        prumer[n-3]=prumer[n-4];
        prumer[n-4]=Reg_vel;
        soucet=0;
        for(n=1; n<6; n++){
              soucet=soucet+prumer[n];
        }
        X=soucet/5;                          // konec plovoucího průměru

        PS(X);                               // volání funkce PS

        WrPortI(PEDDR, &PEDDRShadow, 0x00); // nastavení portu E jako vstupní
        up=BitRdPortI(PEDR,2); // čtení pinu, na kterém je připojeno tlačítko Up
        WrPortI(PEDDR, &PEDDRShadow, 0x00); // nastavení portu E jako vstupní
        down=BitRdPortI(PEDR,3);             // čtení pinu, na kterém je připojeno
                                             //tlačítko Down
        if(up) W=W+0.5;                      //jestliže bylo stisknuto tlačítko Up
                              //zvětší se žádaná veličiny
        if(down) W=W-0.5;                    //jestliže bylo stisknuto tlačítko Down
                                //zmenší se žádaná veličiny
        msDelay (200);

        pom_W=W*10;              // uložení floatové proměnné na intovou
        W_tep=pom_W/10;
        W_des=pom_W%10;
        pom_X=X*10;              // uložení floatové proměnné na intovou
        tep=pom_X/10;
        des=pom_X%10;

  // na displeji je možné zobrazi pouze informace uložené v proměnné typu char
        itoa(tep, p_teplota);   // proto jsou zde převáděny z typu INT na typ CHAR
        itoa(des, p_desetiny);  // na displeji zobrazujeme aktuální teplotu
        itoa(W_tep, p_W);       // požadovanou teplotu
        itoa(W_des, pW_des);    // a akční zásah vyjádřený v procentech
        Akcni=Akcni_Zasah*100;  // vyjádření akčního zásahu v procentech
        itoa(Akcni, p_Akcni);

        // vypisování informací na displej
        Clear();                             // volání funkce Clear
        Display ( "Akt.teplota:" );          // zobrazení Akt.teplota na displej
        Display (p_teplota);
        Display (",");                       // zobrazení čárky na displej
        Display (p_desetiny);
        Display("ßC");                       // zobrazení stupňů celsia na displej
        Line2 ();
        Display ( "Zad.teplota:" );          // zobrazení Zad.teplota na displej
        Display (p_W);
        Display (",");                       // zobrazení čárky na displej
        Display (pW_des);
        Display("ßC");                       // zobrazení stupňů celsia na displej
        Line3 ();
        Display ( "Akcni zasah:" );          // zobrazení Akcni zasah na displej
        Display (p_Akcni);
        Display ( "%" );                     // zobrazení symbolu procent na displej

  // realizace převodu akčního zásahu na ON/OFF (relé s hysterezí)
        if(Akcni_Zasah<3){
              BitWrPortI(PBDR, &PBDRShadow, 0x00, 7); // nastavení nuly na pin 7
                                                      //portu B
```

```
        }                                           // vypnutí indikační Led diody
        if(Akcni_Zasah>5){
              BitWrPortI(PBDR, &PBDRShadow, 0xFF, 7); // nastavení jedničky na pin 7
                                                      //portu B
        }                                           // zapnutí indikační Led diody
        if (BitRdPortI(PBDR,7)){
              Line4();
              Display ( " Kotel:ON" );      // zobrazení Kotel:ON na displej
        }                                   // v případě požadavku na zapnutí kotle
        else {
              Line4();
              Display ( " Kotel:OFF" );     // zobrazení Kotel:OFF na displej
        }                                   // v případě požadavku na vypnutí kotle
    }
}
```